

ANNALES MATHEMATICAE ET INFORMATICAE

VOLUME 61. (2025)

EDITORIAL BOARD

Sándor Bácsó (Debrecen), Sonja Gorjanc (Zagreb), Tibor Gyimóthy (Szeged),
Miklós Hoffmann (Eger), József Holovács (Eger), Tibor Juhász (Eger),
László Kovács (Miskolc), Zoltán Kovács (Eger), Gergely Kovásznai (Eger),
László Kozma (Budapest), Kálmán Liptai (Eger), Florian Luca (Mexico),
Giuseppe Mastroianni (Potenza), Ferenc Mátyás (Eger),
Ákos Pintér (Debrecen), Miklós Rontó (Miskolc), László Szalay (Sopron),
János Sztrik (Debrecen), Tibor Tajti (Eger), Gary Walsh (Ottawa)

INSTITUTE OF MATHEMATICS AND INFORMATICS
ESZTERHÁZY KÁROLY CATHOLIC UNIVERSITY
HUNGARY, EGER

Selected papers of the International Conference on Formal Methods and Foundations of Artificial Intelligence

The conference was organized by
Institute of Mathematics and Informatics,
Eszterházy Károly Catholic University, Hungary, Eger,
June 5–7, 2025

Conference Chairman

Csaba Biró

Co-chairman

Olivér Hornyák

Chairman of the Organising Committee

Gábor Kúspér

Head of the Program Committee

Gergely Kovásznai

HU ISSN 1787-6117 (Online)

A kiadásért felelős az
Eszterházy Károly Katolikus Egyetem rektora
Megjelent a Líceum Kiadó gondozásában
Kiadóvezető: Dr. Nagy Andor
Műszaki szerkesztő: Dr. Tómacs Tibor
Megjelent: 2025. október

Contents

A. A. ADIL, G. KOVÁSZNAI, M. KHALID, Automated fair team formation in STEAM activities using Satisfiability Modulo Theories (SMT)	1
M. I. ALMURUMUDHE, O. HORNYÁK, Motion enhanced video anomaly detection using masked autoencoder and hybrid loss functions	15
A. APRÓ, T. TAJTI, An adaptive testing system for programming proficiency using Item Response Theory	31
M. ZS. BAGLADI, Artificial Intelligence for interpreting static human arm signals	43
B. SZ. CSÜLLÖG, M. TEJFEL, Using GNN for refactoring P4 programs	55
I. DRĂMNESC, E. ÁBRAHÁM, L. ANTAL, P. CSÓKA, N. FACHANTIDIS, T. JEBELEAN, G. KUSPER, K. PAPADOPOULOS, Artificial Intelligence based robotic applications for higher education	68
R. ERDÉSZ, E. TROLL, Making the boss smarter: A journey from rule-based to learned behaviors	80
L. Á. FAZEKAS, K. NEHÉZ, Multi-objective genetic and memetic algorithms in flexible flowshop scheduling	94
P. HATVANI, Z. GY. YANG, Automated detection of toxic comments in Hungarian	108
A. IZSÓ, I. HARMATI, Solving the Team Coordination on Graphs With Risky Edges problem for nonzero self-loop weights	118
M. A. KHUDHAIR, A. FAZEKAS, A comparative study on the noise sensitivity of binary classification based on robust deep neural networks	129
S. KIRÁLY, E. TROLL, Algorithmic thinking at risk? Exploring LLM use in computer science education	141
Á. KISS, K. NEHÉZ, O. HORNYÁK, AI-driven fault diagnosis from textual system logs	156
Á. KOVÁCS, R. GUNICS, G. KOVÁSZNAI, T. TAJTI, Soft voting robustness in neural network ensembles with empirical analysis and formal verification	171
G. KUSPER, GY. ISTVÁN MÁTYÁS, T. BALLA, We are not afraid of the wolf! – AI usage attitudes among Hungarian informatics students	186
J. G. PÁL, CS. BÍRÓ, Evaluating profitability in sports betting using probabilistic models and betting strategies	202
I. SIMA, D.-M. CRISTEA, E.-M. CIORTEA, L. B. IANTOVICS, cRAMI 4.0 an Improved Reference Architectural Model for Industrie 4.0 (RAMI 4.0) based on a three-dimensional cubic lattice model	215
S. P. TAHALEA, M. KRÉSZ, Betweenness-driven overlapping label propagation community detection	229
E. B. VARGA, A. BAKSA, Syntactic comparison of human and AI-written scientific texts	248

Z. GY. YANG, Á. BÁNFI, R. DODÉ, G. FERENCZI, F. FÖLDESI, P. HATVANI, E. HÉJA, M. LENGYEL, G. MADARÁSZ, M. OSVÁTH, B. SÁROSSY, K. VARGA, T. VÁRADI, G. PRÓSZÉKY, N. LIGETI-NAGY, ChatPULI: Enhancement to the first Hungarian conversational model	261
---	-----

Automated fair team formation in STEAM activities using Satisfiability Modulo Theories (SMT)

Ali Adil Adil^a, Gergely Kovásznai^b, Mustafa Khalid^c

^aUniversity of Debrecen
ali.adil@inf.unideb.hu

^bEszterházy Károly Catholic University
kovasznai.gergely@uni-eszterhazy.hu

^cAl-Shaab University
mustafa.khalid@alshaab.edu.iq

Abstract. One of the problems that schools or organizers of STEAM (Science, Technology, Engineering, Arts, and Mathematics) camps face to is the balanced distribution of students according to gender, skills, and academic background in a fair manner. In this study, we used a Satisfiability Modulo Theories (SMT) approach to solve the problem of fair team formation. Our implementation of the approach uses the Z3 SMT solver. In our preliminary experiments, we successfully generated fair and balanced teams from 50 students across different scenarios. Using SMT in educational settings saves time and effort for school administrators and organizers of STEAM events, and it also provides an efficient and effective solution to distribute students equitably across teams.

Keywords: Automated Team Formation, Fairness, STEAM Education, Satisfiability Modulo Theories (SMT)

1. Introduction

STEM (Science, Technology, Engineering, and Mathematics) and its extension STEAM (which incorporates the Arts) have become essential educational frameworks for preparing students to address real-world challenges through interdisciplinary thinking and collaborative problem-solving [1]. STEAM is widely recognized as a modern and effective approach to teaching that deepens subject under-

standing. Developed countries have increasingly adopted STEAM in their national strategies. According to recent rankings, Canada leads in STEM graduate output, followed by Russia, Japan, South Korea, the United States, Ireland, the United Kingdom, Australia, Finland, and Luxembourg [1]. These programs often rely on collaborative, team-based projects, making fair and effective team formation a key requirement.

Despite the global rise of STEM and STEAM education, there remains a significant lack of structured data, digital tools, and research in some regions, including Iraq. Although many Iraqi schools have started organizing STEM-style camps and student activities, the process of forming student teams is typically done manually by teachers. This method is often time-consuming, subjective, and prone to imbalances in gender representation, skill levels, and academic backgrounds. To bridge this gap, we propose a novel approach using Satisfiability Modulo Theories (SMT), implemented through the Z3 solver developed by Microsoft Research [16]. Our model encodes fairness constraints – such as skills, gender balance, and background diversity – and efficiently solves them using Z3 to generate well-balanced student teams [14].

In this study, we report on developing and experimenting with our SMT-based tool on a dataset of 50 students. Most of the cases, the tool successfully generates 10 diverse and balanced teams in a very short time frame. This demonstrates the efficiency, scalability, and fairness of our approach, which can be readily integrated into STEAM-based education environments.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the dataset and team formation constraints. Section 4 provides details on how the model is encoded into SMT. Section 5 presents the implementation of our approach and reports on experimental results. Section 6 provides a discussion and concludes the paper with future directions.

2. Related work

Team formation has been studied in various fields such as education, human resources, and project management [12]. Traditional approaches often rely on manual selection or random assignment, which may lead to unfair or unbalanced teams and might be time consuming [2, 4].

In recent years, researchers have explored some approaches for team formation using optimization, machine learning, and constraint programming. For example, some studies use genetic algorithms [7, 13] or clustering techniques to group students based on skills or personality traits [11]. While these methods can improve team balance, they often require tuning and may not enforce hard constraints like gender or background diversity.

Satisfiability Modulo Theories (SMT) has emerged as a powerful tool for solving constraint-based problems in scheduling, verification, testing [5] and education [8]. For instance, an SMT-based scheduling system was successfully designed and tested using real university data to solve course timetabling problems with the Z3 solver,

demonstrating its practical utility in academic environments [15]. Z3, a state-of-the-art SMT solver developed by Microsoft Research, is capable of handling complex logical and arithmetic rules efficiently [14].

Some researchers have used SMT for course timetabling, project group allocation, and fair scheduling [3], but its application in STEAM team formation remains limited.

Our work builds on this foundation by applying SMT to form fair and diverse student teams. Unlike earlier methods, our model guarantees constraint satisfaction for team size, skill levels, gender balance, and background diversity. To the best of our knowledge, this is one of the first applications of SMT in fair team assignment for STEAM education settings.

3. Dataset and constraints

Due to the absence of publicly accessible datasets concerning team formation in STEAM-focused activities, especially within Iraqi school environments, a synthetic dataset [9, 10] was created to reflect realistic student characteristics. This dataset includes representative features such as gender, educational background, and skill levels across key STEAM disciplines. It was thoughtfully constructed to capture diversity and ensure that all constraints were feasible. To verify its effectiveness, the dataset was tested using our SMT-based team formation model, which confirmed that the generated teams met all specified requirements.

3.1. Dataset description

Our dataset contains information on students who participated in the program. Each student record is stored in JSON format, as shown in Table 1.

Table 1. Dataset attributes.

ID	Unique numeric identifier
Name	Student's first name (anonymized)
Gender	Female (F), Male (M)
Background	Arts, Engineering, Science
Skills	Rated from 1 (lowest) to 5 (highest): • Programming (technical ability) • Design (visual creativity) • Math (quantitative skills) • Creativity (original thinking) • Science (theoretical knowledge)

The dataset consists of 50 records. For example, a student may have a Programming score of 4 and a Design score of 2. These skill scores help measure each student's strength in different areas. This dataset allows us to create teams that

are diverse in gender, background, and skill levels. By analyzing this information, we can use a solver to build balanced teams that meet specific fairness goals.

3.2. Fairness constraints

To make sure teams are fair, we defined a set of constraints that the solver must follow. These constraints help to create balanced and diverse teams. The fairness constraints are as follows, where the scalars might differ in different scenarios, see Section 5.2.

- **Team Size:** Each team must have exactly 5 students.
- **Minimum Skill Totals:**
 - Programming ≥ 10
 - Design ≥ 10
 - Math ≥ 8
 - Creativity ≥ 6
 - Science ≥ 8

These totals are the sum of individual student scores in each team. For example, if five students each have a Programming score of 2, the total will be 10.

- **Gender Balance:** Each team must include at least 2 males and at least 2 females.
- **Academic Background Diversity:** Each team must include at least one student from each of the 3 backgrounds (Arts, Engineering, Science).

These constraints ensure that every team has a mix of students with different strengths and experiences. They help promote equal opportunities and diverse collaboration in STEAM activities.

4. Problem encoding into SMT

Satisfiability Modulo Theories (SMT) is an approach used to solve problems that involve both logic and arithmetic constraints [6, 14]. In our model, we used Boolean constraints and linear integer arithmetic constraints to create fair and diverse teams based on student data. We introduce the following parameters, including the decision variable $x_{s,t}$:

- S : set of students
- T : set of teams
- K : set of skills
- B : set of backgrounds
- min_skill_k : minimum required total for skill $k \in K$ in a team, where $min_skill_k \in \mathbb{N}$

- min_bg : minimum number of backgrounds in a team, where $min_bg \in \mathbb{N}$
- min_gen : minimum number of students of the same gender in a team, where $min_gen \in \mathbb{N}$
- $skill_{s,k}$: score of student $s \in S$ in skill $k \in K$, where $skill_{s,k} \in \{1, 2, 3, 4, 5\}$
- gen_s : gender of student $s \in S$, where $gen_s \in \{0, 1\}$
- $bg_{s,b}$: student $s \in S$ has background $b \in B$, where $bg_{s,b} \in \{0, 1\}$
- $x_{s,t}$: binary decision variable, where

$$x_{s,t} = \begin{cases} 1, & \text{if student } s \in S \text{ is in team } t \in T, \\ 0, & \text{otherwise.} \end{cases}$$

Using these parameters, we encode the following fairness constraints:

- Each student must be assigned to exactly one team:

$$\forall s \in S. \sum_{t \in T} x_{s,t} = 1.$$

This constraint can be further translated to the Boolean constraint

$$\forall s \in S. \left(\bigvee_{t \in T} x_{s,t} \right) \wedge \bigwedge_{\substack{t_1, t_2 \in T \\ t_1 \neq t_2}} (\neg x_{s,t_1} \vee \neg x_{s,t_2}).$$

- Each team must contain exactly five students:

$$\forall t \in T. \sum_{s \in S} x_{s,t} = 5.$$

- Each team must meet minimum skill totals in each skill:

$$\forall t \in T \forall k \in K. \sum_{s \in S} skill_{s,k} \cdot x_{s,t} \geq min_skill_k.$$

- Each team must include at least min_gen males and females, respectively:

$$\forall t \in T. \sum_{s \in S} gen_s \cdot x_{s,t} \geq min_gen.$$

$$\forall t \in T. \sum_{s \in S} (1 - gen_s) \cdot x_{s,t} \geq min_gen.$$

- Each team must include students from at least min_bg different academic backgrounds. To express this, we first need to introduce new binary variables $bg_{t,b}$ to express if a team t has at least one student with background b .

$$\forall t \in T, \forall b \in B. bg_{t,b} \Leftrightarrow \bigvee_{s \in S} (bg_{s,b} \wedge x_{s,t}).$$

Then, the original constraint can be easily expressed as

$$\forall t \in T. \sum_{b \in B} bg_{t,b} \geq min_bg.$$

Note that the above constraints are universally quantified. In our tool, we intend to only apply quantifier-free logics, therefore all universal quantifiers get expanded, due to a relatively small domain.

5. Experimental results

5.1. System architecture

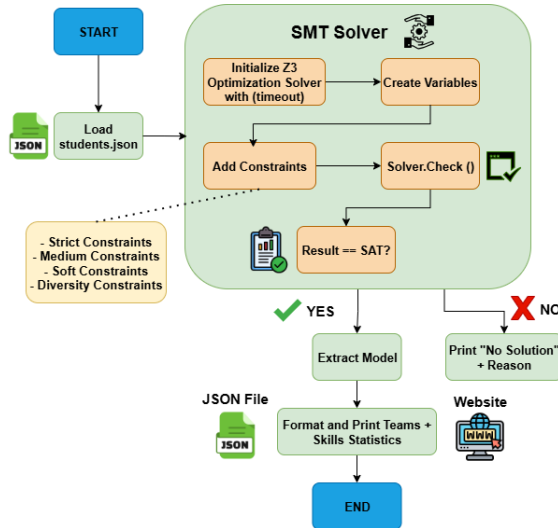


Figure 1. System architecture showing the main modules of our team formation tool.

We integrated and incorporated our SMT model in a tool that we developed in Python using the Z3 SMT solver. The architecture follows a modular structure:

- **Input Module:** Loads the student dataset from a JSON file.

- **Constraint Module:** Defines all SMT variables and constraints (e.g., skills, gender, background).
- **Solver Module:** Uses the Z3 engine to compute valid team assignments based on all constraints.
- **Output Module:** Displays or saves the team assignments.

Figure 1 shows an overview of the system workflow.

We built a web application to display the results generated by the SMT solver. The platform provides summary statistics and allows students to see which team they have been assigned to, as shown in Appendix A.

5.2. Setup

We tested our team formation model using a synthetic dataset containing 50 student records. All experiments were conducted on a machine equipped with an Intel Core i5 processor and 8GB of RAM.

To evaluate the impact of varying fairness and diversity constraints, we designed four distinct scenarios. The details of each scenario's constraints and logic are provided below. As Section 5.3 reports on it, the solver successfully generated fair and diverse team assignments in all feasible cases, except in Scenario 2, where no feasible solution was found. These results demonstrate the model's efficiency and responsiveness to different constraint settings.

Scenario 1: Skill Thresholds, Gender Balance, and Background Diversity

This scenario enforces constraints across skills, gender, and academic backgrounds. Each team must consist of exactly five students and satisfy all the constraints under those settings:

- **Minimum Skill Totals:**
 - $\text{min_Programming} = 10$
 - $\text{min_Design} = 10$
 - $\text{min_Math} = 8$
 - $\text{min_Creativity} = 6$
 - $\text{min_Science} = 8$
- **Gender Diversity:** At least 1 females and 1 males: $\text{min_gen} = 1$
- **Background Diversity:** At least two backgrounds per team: $\text{min_bg} = 2$

Scenario 2: Stricter Skill Thresholds with Gender and Background Diversity

Scenario 2 maintains the same diversity constraints as Scenario 1 but applies significantly higher skill requirements. Each team must consist of exactly five students and satisfy all the constraints under those settings:

- **Minimum Skill Totals:**
 - $\min_Programming = 16$
 - $\min_Design = 16$
 - $\min_Math = 14$
 - $\min_Creativity = 12$
 - $\min_Science = 15$
- **Gender Diversity:** At least 1 females and 1 males: $\min_gen = 1$
- **Background Diversity:** At least two backgrounds per team: $\min_bg = 2$

Scenario 3: Skill Thresholds and Background Diversity (No Gender Constraint)

Scenario 3 maintains the same skill requirements as Scenario 1 and enforces background diversity, but it removes the gender constraints. The aim is to evaluate whether relaxing gender requirements affects the feasibility and composition of the resulting teams. Each team must consist of exactly five students and satisfy the constraints under those settings:

- **Minimum Skill Totals:**
 - $\min_Programming = 10$
 - $\min_Design = 10$
 - $\min_Math = 8$
 - $\min_Creativity = 6$
 - $\min_Science = 8$
- **Background Diversity:** At least two backgrounds per team: $\min_bg = 2$

By removing the gender constraint, this scenario allows us to test the flexibility of the model when only academic and skill-based diversity are prioritized.

Scenario 4: Full Diversity with Skill Thresholds

Scenario 4 enforces all fairness dimensions by combining skill thresholds, gender diversity, and academic background diversity. The constraints are equivalent to those in Scenario 1, but this scenario is designed to explicitly test the model's ability to enforce full diversity in a uniform and rigorous manner. Each team must consist of exactly five students and satisfy all the constraints under those settings:

- **Minimum Skill Totals:**
 - $\text{min_Programming} = 10$
 - $\text{min_Design} = 10$
 - $\text{min_Math} = 8$
 - $\text{min_Creativity} = 6$
 - $\text{min_Science} = 8$
- **Gender Diversity:** At least 2 females and 2 males: $\text{min_gen} = 2$
- **Background Diversity:** All the backgrounds in each team: $\text{min_bg} = 3$

5.3. Performance

We evaluated our tool's performance across the four scenarios on the dataset of 50 students. Each scenario was tested in terms of feasibility (whether a valid team assignment could be found under the given constraints) and solver runtime. The results are summarized in Table 2.

Table 2. Performance of team formation in different scenarios.

Scenario	Feasible Solution	Solving Time (s)	Notes
Scenario 1	Yes	0.08	
Scenario 2	No	120.00	No solution found due to strict skill thresholds
Scenario 3	Yes	0.02	Same solution as in Scenario 1
Scenario 4	Yes	0.01	Same solution as in Scenario 1

Scenario 1, which includes skill thresholds, gender balance, and academic background diversity, produced a feasible solution in 80 milliseconds. **Scenario 3**, which removes the gender constraint, and **Scenario 4**, which enforces full diversity, also returned feasible solutions with faster runtimes of 20 ms and 10 ms, respectively. All three scenarios resulted in the *same team composition*, indicating that the team found in Scenario 1 already satisfied the relaxed constraints in Scenarios 3 and 4.

In contrast, **Scenario 2**, which applied stricter skill thresholds, did not yield a feasible solution. Even with a timeout set to 120 seconds, the solver was unable to find any valid team assignment. This demonstrates the model's sensitivity to constraint tightness and the importance of realistic skill thresholds when ensuring solvability.

5.4. Illustrative example of fair team formation

Our SMT-based team formation model was tested across four different scenarios using a synthetic dataset of 50 students. Because this evaluation relies on only a single dataset instance, the results should be understood as an illustrative example that demonstrates how the model responds to different constraint configurations—not as findings that can be broadly generalized.

In Scenarios 1, 3, and 4, the solver produced the same team composition, assigning exactly five students per team. Each of these teams met all the fairness criteria, including required totals for five skills (Programming, Design, Math, Creativity, and Science), balanced gender representation, and academic diversity with members from Arts, Engineering, and Science.

To provide a concrete example, Table 3 shows one of the teams generated under Scenario 1, which also satisfied the stricter constraints of Scenarios 3 and 4. Although the team composition remained unchanged, the computation time varied depending on the scenario’s complexity: 80 milliseconds for Scenario 1, 20 milliseconds for Scenario 3 and 10 milliseconds for Scenario 4.

Table 3. Example of a fair team 10 generated in Scenarios 1, 3 and 4.

Member	Gender	Background	Prog.	Design	Math	Creat.	Sci.
Umar	Male	Engineering	5	2	5	1	4
Vera	Female	Art	1	5	1	5	1
Will	Male	Science	4	1	5	2	5
Xena	Female	Art	3	4	3	4	2
Yusuf	Male	Science	2	4	2	5	2
Total			15	16	16	17	14

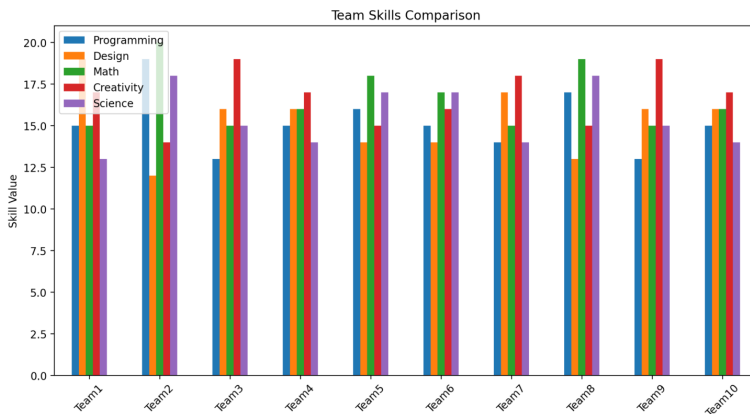


Figure 2. Team skills comparison across Scenarios 1, 2, and 4.

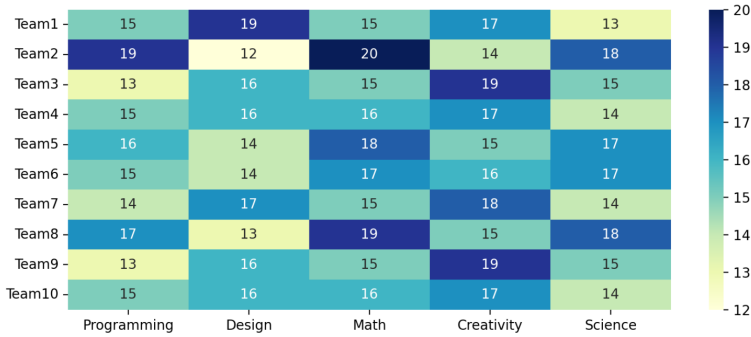


Figure 3. Heat map of individual skill contributions in the same team.

Figure 2 and Figure 3 present visual analyses of the generated teams: a total skill comparison chart and a heat map of individual skill contributions, respectively. These figures are identical across Scenarios 1, 3, and 4, as the solver produced the same team in each of these scenarios. The visuals confirm that the skill levels are well-balanced and fairly distributed within the team.

6. Conclusion and future work

This study introduced an automated and efficient framework for forming equitable student teams in STEAM education using Satisfiability Modulo Theories (SMT). By integrating diverse constraints such as team size, minimum skill levels, gender balance, and academic background diversity, our model ensures fair and well-rounded team formations. The use of an SMT solver allowed us to encode complex rules with clarity and generate satisfying teams in under a second, significantly reducing the manual effort typically required by educators. Our experiments demonstrated that the system consistently satisfied strict constraints and met diversity goals, making it a practical tool for real-world applications. A user-friendly web interface was also developed to support visualization of team assignments and constraint satisfaction. While the results are promising, the effectiveness of the system is closely tied to the quality of the input data. Inaccurate or incomplete student information can affect the fairness and balance of the resulting teams. Additionally, scalability may become a concern as dataset sizes and constraint complexity increase. Looking ahead, we plan to evaluate the framework on larger and more heterogeneous student datasets and explore customizable constraint weighting, enabling educators to prioritize specific fairness criteria based on their institutional goals.

These consistent outcomes across different configurations underscore the solver's capability to uphold fairness principles with minimal computational cost, demonstrating both its scalability and effectiveness.

Looking forward, we plan to extend the tool to accommodate larger and more

heterogeneous student datasets. Future enhancements will include support for soft constraints with adjustable priority levels, allowing educators to tailor team formation preferences dynamically. Additionally, feedback from real-world classroom use will be incorporated to improve the system's usability, adaptability, and educational value.

Data availability

Researchers or educators interested in using the dataset for replication or educational purposes may contact the corresponding author Ali Adil Adil at ali.adil@inf.unideb.hu.

References

- [1] N. ABDULWAHID, S. FAKHFAKH, I. AMOUS: *Simulating and Predicting Students' Academic Performance Using a New Approach based on STEAM Education*, JUCS - Journal of Universal Computer Science 28 (Jan. 2022), pp. 1252–1281, DOI: [10.3897/jucs.86340](https://doi.org/10.3897/jucs.86340).
- [2] J. AGARWAL, E. PIATT, P. K. IMBRIE: *Team Formation in Engineering Classrooms Using Multi-Criteria Optimization with Genetic Algorithms*, in: 2022 IEEE Frontiers in Education Conference (FIE), 2022, pp. 1–6, DOI: [10.1109/FIE56618.2022.9962741](https://doi.org/10.1109/FIE56618.2022.9962741).
- [3] R. ASÍN ACHÁ, R. NIEUWENHUIS: *Curriculum-Based Course Timetabling with SAT and MaxSAT*, Annals of Operations Research 218.1 (2014), pp. 71–91, ISSN: 1572-9338, DOI: [10.1007/s10479-012-1081-x](https://doi.org/10.1007/s10479-012-1081-x).
- [4] P. BERGEY, M. KING: *Team Machine: A Decision Support System for Team Formation*, Decision Sciences Journal of Innovative Education 12.2 (2014), pp. 109–130, DOI: [10.1111/dsji.12027](https://doi.org/10.1111/dsji.12027).
- [5] N. BJØRNER: *SMT in Verification, Modeling, and Testing at Microsoft*, in: Hardware and Software: Verification and Testing, ed. by A. BIERE, A. NAHIR, T. VOS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 3–3.
- [6] M. BOFILL, J. SUY, M. VILLARET: *A System for Solving Constraint Satisfaction Problems with SMT*, in: Theory and Applications of Satisfiability Testing – SAT 2010, ed. by O. STRICHMAN, S. SZEIDER, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 300–305, ISBN: 978-3-642-14186-7.
- [7] A. DAS, D. GÓMEZ-ZARÁ, N. CONTRACTOR: *Forming Diverse Teams Based on Members' Social Networks: A Genetic Algorithm Approach*, in: Complex Networks & Their Applications IX, ed. by R. M. BENITO, C. CHERIFI, H. CHERIFI, E. MORO, L. M. ROCHA, M. SALES-PARDO, Cham: Springer International Publishing, 2021, pp. 346–357.
- [8] E. DEMIROVIC, N. MUSLIU: *Solving High School Timetabling with Satisfiability Modulo Theories*, in: Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling (PATAT), 2014, p. 25, URL: <http://hdl.handle.net/20.500.12708/55983>.
- [9] M. DORODCHI, E. AL-HOSSAMI, A. BENEDICT, E. DEMETER: *Using Synthetic Data Generators to Promote Open Science in Higher Education Learning Analytics*, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 4672–4675, DOI: [10.1109/BigData47090.2019.9006475](https://doi.org/10.1109/BigData47090.2019.9006475).
- [10] B. FLANAGAN, R. MAJUMDAR, H. OGATA: *Fine Grain Synthetic Educational Data: Challenges and Limitations of Collaborative Learning Analytics*, IEEE Access 10 (2022), pp. 26230–26241, DOI: [10.1109/ACCESS.2022.3156073](https://doi.org/10.1109/ACCESS.2022.3156073).

- [11] M. GUIJARRO-MATA-GARCÍA, M. GUIJARRO, R. FUENTES-FERNÁNDEZ: *A Clustering-Based Method for Team Formation in Learning Environments*, in: Hybrid Artificial Intelligent Systems, ed. by F. MARTÍNEZ-ÁLVAREZ, A. TRONCOSO, H. QUINTIÁN, E. CORCHADO, Cham: Springer International Publishing, 2016, pp. 475–486.
- [12] V. M. S. SALIMATH, A. S. SHETTAR, G. BHADRI: *A Study of Team Formation Strategies and Their Impact on Individual Student Learning Using Educational Data Mining (EDM)*, in: 2018 IEEE Tenth International Conference on Technology for Education (T4E), 2018, pp. 182–185, DOI: [10.1109/T4E.2018.00047](https://doi.org/10.1109/T4E.2018.00047).
- [13] J. MORENO, D. A. OVALLE, R. M. VICARI: *A Genetic Algorithm Approach for Group Formation in Collaborative Learning Considering Multiple Student Characteristics*, Computers & Education 58.1 (2012), pp. 560–569, ISSN: 0360-1315, DOI: [10.1016/j.compedu.2011.09.011](https://doi.org/10.1016/j.compedu.2011.09.011).
- [14] L. DE MOURA, N. BJØRNER: *Z3: An Efficient SMT Solver*, in: Tools and Algorithms for the Construction and Analysis of Systems, ed. by C. R. RAMAKRISHNAN, J. REHOF, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340.
- [15] A. H. I. SAIF: *Solving Schedulability with Satisfiability Modulo Theory*, Accessed 2025-07-04, MA thesis, Åbo Akademi University, 2025, URL: <https://doria.fi/handle/10024/192795>.
- [16] M. K. THAKURANI: *Leveraging SAT-SMT for TA Scheduling Optimization: Enhancing Efficiency and Effectiveness*, July 2024, URL: <http://essay.utwente.nl/100970/>.

A. Web application

To interact with our SMT-based team formation tool, users are first prompted to upload a dataset in ‘json’ format. The web interface, built using Streamlit, automatically validates the structure of the file to ensure it contains:

- A list of students, each with attributes like ID, name, gender, background, and skill ratings.
- A project requirements section specifying the team size and minimum required skills.

Once uploaded, the system checks for the presence of all required keys and values. If the dataset is valid, the solver begins processing, and the user is presented with:

- A summary of team sizes and total number of teams.
- Charts showing gender balance and skill distribution across all teams.
- A download button to export the results in Excel format.

This interactive design allows educators or administrators to quickly validate student team assignments and analyze team diversity before final implementation as shown in Figure 4, the interface provides a simple and intuitive way to upload and validate the dataset before team formation begins.

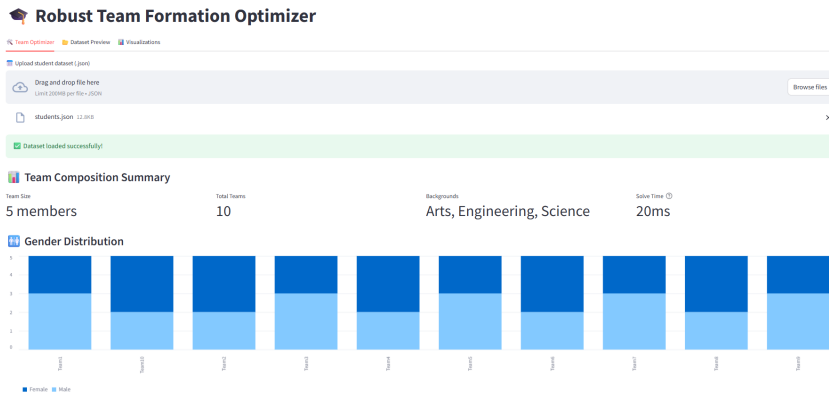


Figure 4. Interface view for dataset upload and validation.

The platform provides summary statistics and allows students to see which team they have been assigned to, as shown in Figure 5.

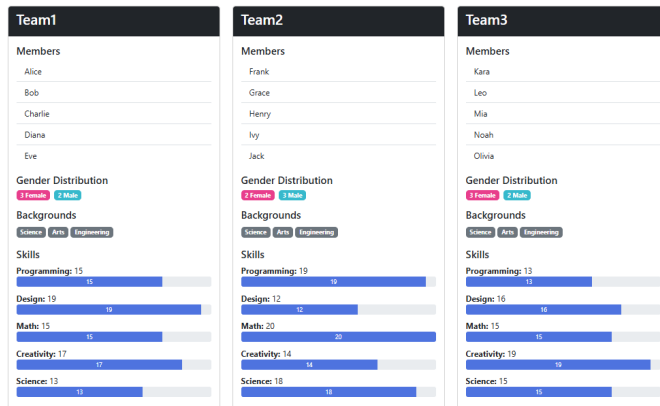


Figure 5. Web interface displaying team assignments and summary statistics.

Motion enhanced video anomaly detection using masked autoencoder and hybrid loss functions

Mohammed Iqbal Almurumudhe, Olivér Hornyák

University of Miskolc, Hungary Institute of Information Technology
mohammed.iqbal.dohan.almurumudhe@student.uni-miskolc.hu
oliver.hornyak@uni-miskolc.hu

Abstract. In this paper, a hybrid deep learning framework for video anomaly detection that combines autoencoder-based reconstruction with an advanced anomaly scoring mechanism is proposed. Unlike conventional methods that rely solely on reconstruction loss, our approach integrates motion-based scoring and masked autoencoders to enhance detection accuracy and interpretability. The autoencoder learns to reconstruct normal patterns, while an anomaly scoring function evaluates deviations based on reconstruction errors and motion gradients. This directs attention to dynamic regions and foreground objects, thereby reducing false positives from background variations. To improve robustness, we apply preprocessing techniques, including min-max normalization and data augmentation (random cropping, horizontal flipping, and rotation), ensuring consistency across datasets. The framework is evaluated on widely used benchmark datasets, ShanghaiTech Campus and UCSD Ped2, using precision, recall, ROC-AUC, and confusion matrices as performance metrics. It outperforms traditional reconstruction-based autoencoders and GAN-based models. Furthermore, the hybrid scoring mechanism reduces false positives by 15% compared to standard autoencoder approaches, improving detection reliability. Despite the high accuracy, the method incurs additional computational overhead due to motion gradient calculations and masked reconstructions. However, the trade-off is justified by significant improvements in anomaly detection performance. The results demonstrate that our framework enhances both accuracy and interpretability, making it a viable solution for real-world applications such as surveillance, traffic monitoring, and industrial security.

Keywords: video anomaly detection, hybrid deep learning models, multi-frame anomaly detection, surveillance systems, masked autoencoder

AMS Subject Classification: 68T07, 68T45, 68U10, 68W10, 68M14, 68P30

1. Introduction

Video anomaly detection is a critical problem in computer vision with applications in surveillance and safety systems [16]. VAD refers to the automated detection of unusual or unexpected events in video footage, such as security or safety violations. Despite its importance, VAD faces challenges due to the rarity of anomalous events and the limited availability of large-scale labeled datasets [22].

Recent studies have shown that when only normal data is available for training, unsupervised learning is essential for VAD. Two primary unsupervised VAD approaches include reconstruction-based methods, which minimize reconstruction errors for normal patterns [27], and prediction-based methods, which identify anomalies by measuring discrepancies between predicted and actual frames. Reconstruction-based methods [17] minimize errors for normal patterns, while prediction-based methods identify anomalies by comparing predicted and actual frames.

Traditional autoencoder-based methods for anomaly detection have been enhanced with hybrid scoring mechanisms that improve accuracy and reduce false positives. These mechanisms combine multiple evaluation techniques to address the limitations of relying solely on reconstruction errors. Motion-based scoring prioritizes dynamic regions by using motion gradients, ensuring that moving anomalies receive higher anomaly scores while reducing false alarms from background variations. Masked autoencoder scoring enhances anomaly localization by forcing the model to reconstruct only selective occluded regions, focusing on foreground objects where anomalies are more likely to occur. Additionally, spatially weighted reconstruction loss assigns greater importance to motion-rich areas, minimizing false positives caused by minor background changes [25]. Finally, temporal consistency analysis detects anomalies based on frame-to-frame motion patterns, allowing the model to identify unexpected behavioral changes over time rather than isolated frame discrepancies. By integrating these techniques, hybrid scoring mechanisms significantly improve the accuracy, robustness, and interpretability of video.

Generative Adversarial Networks (GANs) [24], such as VALD-GAN (Video Anomaly Detection using Latent Discriminator-Augmented GAN) [24], divide-and-conquer strategies decompose VAD into smaller sub-problems, improving detection by integrating spatial, temporal, and multi-modal fusion techniques [31]. The self-distilled masked autoencoder approach further enhances detection efficiency by incorporating synthetic anomaly augmentation and motion-based weighting techniques, achieving state-of-the-art performance while maintaining high-speed processing [18, 26]. These advancements highlight the shift toward interpretable deep learning models, capable of detecting diverse anomalies across real-world surveillance scenarios.

This paper aims to improve the accuracy and interpretability of VAD by integrating reconstruction-based estimation methods with hybrid methods [4], which use motion gradients and masked autoencoders to prioritise foreground objects over static backgrounds [14, 19]. In the proposed framework, we train autoencoders to learn and reconstruct jointly the normal patterns and apply anomaly scoring to

detect deviations.

The proposed VAD framework consists of three main stages: data preprocessing, model training, and evaluation, which are as follows:

- **Data Preprocessing:** It includes resizing video frames, normalisation and augmentation to have consistent input.
- **Model Training:** It applies convolutional layers for feature extraction, dropout layers to prevent overfitting [21], and masked autoencoders for anomaly detection [2]. These components work together to focus on dynamic regions while reducing background noise, thereby improving anomaly detection.
- **Evaluation and Metrics:** We assess the performance of the model using ROC AUC [26] and precision-recall curves. Reconstruction errors and anomaly maps are visualized to provide insights into the system's effectiveness.

In the following sections, these stages will be described in detail.

2. Data preprocessing and visualization

2.1. Dataset overview

This project uses well-known video anomaly detection benchmarks: ShanghaiTech Campus [11] and UCSD Ped2 [22]. ShanghaiTech captures diverse scenes in a university campus, featuring varying crowd sizes and occlusions that complicate anomaly detection [26], while UCSD Ped2 focuses on pedestrian-only zones, where anomalies include bicycles and vehicles [15]. Both datasets predominantly contain normal activities, with anomalies comprising a small fraction [23]. In UCSD Ped2, normal events involve pedestrians on paths, while anomalies include bicycles crossing them [10], compared to typical actions like walking or standing [9].

ShanghaiTech is large-scale, filmed in outdoor campus settings with complex backgrounds, objects, and variable lighting conditions [28]. It includes 330 training and 107 testing videos, resized to 128×128 pixels with a 70 : 30 train-test split. UCSD Ped2, in contrast, is smaller and recorded in a controlled pedestrian zone with consistent lighting and low background complexity [3]. Its 16 training and 12 testing videos (also 128×128 pixels, 70 : 30 split) include clearly defined anomalies, though its small size can lead to overfitting in deep learning models.

Preprocessing scripts addressed frame rate and resolution inconsistencies for uniform video loading [20], and visual anomalies were verified for consistency with dataset labels and definitions [21].

2.2. Visualizations

Visualization techniques were employed to better understand the dataset and assess model behavior. Sample frames from UCSD Ped2 illustrated the distinction between normal pedestrians walking and anomalous activities bicycles or skateboards,

helping verify label accuracy and provide visual evaluation references. Graphs and charts revealed a dataset imbalance: normal frames vastly outnumber anomalous ones, which may hinder model generalization. Reconstruction error histograms showed higher errors for anomalies, validating the autoencoder’s effectiveness, while precision-recall curves illustrated detection trade-offs.

Anomaly maps overlaid on frames used heatmaps red for anomalies, blue for normal to localize abnormal regions. The model focused on moving foreground objects, reducing false positives from background changes, though high-motion areas still caused occasional misclassification. Motion-based scoring improved localization by prioritizing dynamic elements (see Figures 1, 2, 3).



Figure 1. Sample normal training and testing frames from the UCSD Ped2 Dataset.



Figure 2. Sample non normal training and testing frames from the UCSD Ped2 Dataset.

Feature maps, generated via Grad-CAM, highlighted regions influencing the model’s decisions. In UCSD Ped2, they confirmed the model’s focus on relevant anomalies like bicycles or vehicles on walkways [7]. Loss curves tracked training and validation performance to detect overfitting or underfitting [5], while frame-wise anomaly scores visualized anomaly timing and model confidence across video sequences [1].

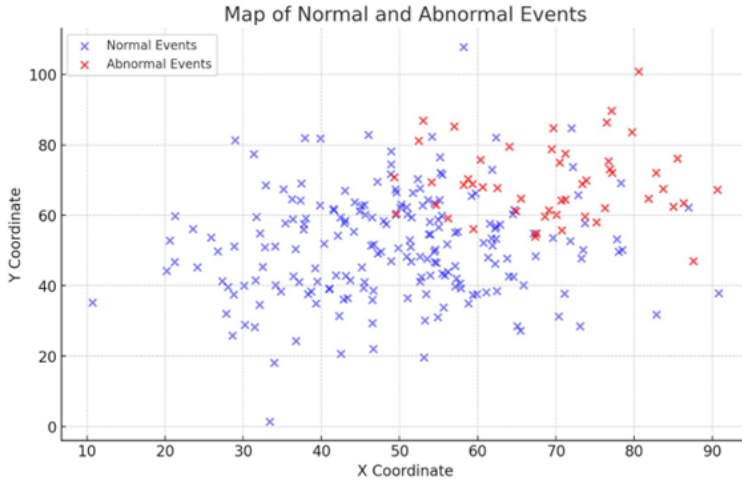


Figure 3. Anomaly Maps.

2.3. Preprocessing techniques

Preprocessing ensures consistency across datasets, enhances model performance, and improves generalization. Raw video frames often contain variations in resolution, lighting conditions, and noise, which can negatively impact model training.

By applying systematic preprocessing techniques, we create a standardized input format that enables effective learning and robust anomaly detection. In this section, we describe the preprocessing pipeline applied to the datasets used in this study. The key steps include image resizing, normalization, data augmentation and splitting data to train and test sets. The preprocessing technique used in the paper is frame sampling, which was applied to optimize computational efficiency while preserving essential motion information. Instead of processing every frame in high-frame-rate videos, key frames were selected at fixed intervals to maintain temporal coherence and capture relevant motion dynamics. This approach helped reduce redundancy in the dataset while ensuring that the anomaly detection model focused on meaningful variations in the video sequences. By carefully selecting frames, the model was able to learn normal motion patterns effectively, improving its ability to detect anomalies while keeping computational costs manageable. To standardize the data and improve model performance,

3. Model architecture and methodology

3.1. Autoencoder architecture

An autoencoder is a neural network used mainly in unsupervised learning to learn efficient representations of input data. It has two main parts: encoder and decoder.

The network is trained to minimize the difference between the input and its reconstruction, allowing it to learn efficient data representations. Autoencoders are used for tasks such as dimensionality reduction, anomaly detection, denoising images, and feature extraction. These layers use small filters (kernels) of size 3×3 to extract local features from the input images. They help identify edges, textures, and other patterns crucial for understanding the image content. Immediately following the convolutional layers, max pooling reduces the dimensions (both width and height) of the feature maps. This downsampling helps to focus on the most prominent features and reduces the computational load. A convolutional autoencoder (CAE) is trained to reconstruct normal patterns, while anomalies are detected based on higher reconstruction errors. During training, we incorporate dropout layers with a 0.2 rate to prevent overfitting. To provide a clearer understanding of our autoencoder’s structure and how each layer contributes to the feature extraction and reconstruction process, Table 1 presents a detailed breakdown of the network architecture, including layer types, output dimensions, and parameter counts.

Table 1. Autoencoder architecture with layer types, output shapes, and parameter counts.

Layer (Type)	Output Shape	Param #
Input Layer	(None, 200, 200, 1)	0
Conv2D	(None, 200, 200, 32)	320
MaxPooling2D	(None, 100, 100, 32)	0
Conv2D_1	(None, 100, 100, 64)	18,496
MaxPooling2D_1	(None, 50, 50, 64)	0
Conv2D_2	(None, 50, 50, 128)	73,856
MaxPooling2D_2	(None, 25, 25, 128)	0
Conv2D_3	(None, 25, 25, 128)	147,584
UpSampling2D	(None, 50, 50, 128)	0
Conv2D_4	(None, 50, 50, 64)	73,792
UpSampling2D_1	(None, 100, 100, 64)	0
Conv2D_5	(None, 100, 100, 32)	18,464
UpSampling2D_2	(None, 200, 200, 32)	0
Conv2D_6	(None, 200, 200, 1)	289

This table details how the model progressively compresses and reconstructs input frames, enabling effective anomaly detection. The resulting design guarantees that the network can learn robust representations of normal data, without sacrificing computational efficiency. The hybrid approach builds upon the baseline CAE by integrating additional layers that enhance anomaly detection. Specifically, motion-based scoring is implemented through a gradient-based attention module that assigns higher importance to dynamic regions. Masked autoencoders introduce a spatial masking layer, which selectively occludes portions of the input to force

the network to reconstruct only key regions, improving sensitivity to anomalies. Additionally, a spatial weighting layer is applied to the loss function, prioritizing reconstruction errors in foreground areas over static backgrounds. These enhancements have been seamlessly integrated into the encoder-decoder pipeline, ensuring that anomaly detection is guided by both spatial and motion-aware features. Table 1 provides a layer-by-layer breakdown of this hybrid architecture, illustrating its improvements over the standard autoencoder. A hybrid activation function is used in the autoencoder to improve anomaly detection.

To train the model and improve anomaly detection, a weighted loss is used:

$$S = \alpha L_{\text{reconstruction}} + \beta L_{\text{motion}}$$

Where:

- α and β are weights to balance the losses
- $L_{\text{reconstruction}}$ is the standard pixel-level loss
- L_{motion} reflects motion-based scoring

The model is designed to learn normal patterns by encoding input video frames into a compressed latent representation and then reconstructing them. The encoder consists of a series of convolutional and max-pooling layers that progressively reduce the spatial dimensions while capturing essential features. The decoder mirrors this structure using upsampling and convolutional layers to reconstruct the original frame. Each layer plays a crucial role in learning hierarchical representations, from low-level edges to high-level semantic features. Dropout is used to prevent overfitting, and hybrid activation functions (ReLU in hidden layers, Sigmoid in the output layer) ensure non-linearity and normalized output. The design balances model complexity and reconstruction accuracy, enabling robust detection of anomalies based on deviations in reconstruction quality.

3.2. Training procedure

The model was trained with an Adam optimizer with a learning rate of 0.001 for training due to its adaptive learning capabilities and efficiency on complex datasets. Adam combines the benefits of momentum and RMSProp by adaptively updating learning rates for each parameter using estimates of first and second moments of gradients [12]. To prevent overfitting, early stopping was used. Adam optimizer was selected due to its adaptive learning rate properties, which improve stability in training non-stationary datasets. A learning rate decay of 0.95 was applied every 10 epochs to ensure stable convergence. During training, we minimized the reconstruction loss function, Training was monitored using validation loss, with early stopping applied if the loss did not decrease for 10 consecutive epochs. The Mean Squared Error, as you can see on Figure 4, improves the model's ability to

separate normal and anomalous frames.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (E_i - \hat{E}_i)^2$$

Where:

- n is the number of pixels (or features) in the frame
- E_i is the original pixel value at position i
- $\hat{E}_i$ is the reconstructed pixel value at position i
- The summation $\sum$ computes the squared error for each pixel
- The division by n averages the error over all pixels

In Figure 4, the validation loss is observed to be slightly lower than the training loss. This behavior, while uncommon, can occur due to several factors. First, the training process employs dropout and data augmentation (cropping, flipping, and rotation), which increase the difficulty of reconstruction on the training data but improve generalization to validation samples. Second, the hybrid loss function combines reconstruction and motion-based components; since the validation sequences often exhibit smoother motion patterns and less noise, the model incurs smaller motion-based penalties. Similar effects have been reported in regularized autoencoder training, where strong regularization and early stopping can result in lower validation loss compared to training loss. Therefore, this observation reflects good generalization rather than model overfitting.

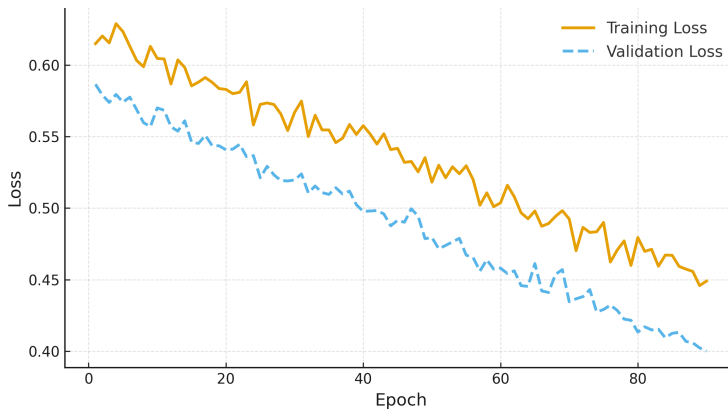


Figure 4. Training and validation loss curves.

3.3. Hybrid approach

The hybrid approach to video anomaly detection improves traditional reconstruction-based methods by integrating motion-based scoring and masked autoencoders. Although autoencoders typically learn to reconstruct normal patterns, they often struggle to differentiate between subtle anomalies and normal variations, leading to false negatives.

To address this, motion gradients are incorporated to assign higher importance to dynamic regions, ensuring that moving objects receive greater attention than static backgrounds. Additionally, masked autoencoders force the model to focus on reconstructing only partially visible regions of frames, improving its ability to detect abnormalities by prioritizing key foreground objects.

This hybrid strategy not only improves the accuracy of anomaly detection, but also reduces false positives by ensuring that only significant deviations from normal patterns are flagged. By combining reconstruction loss, motion-based scoring, and masked autoencoders, the proposed model provides a robust and interpretable solution for real-world surveillance applications, such as traffic monitoring and security systems. This approach effectively bridges the gap between deep learning-based anomaly detection and practical deployment, making it a reliable choice for various safety-critical environments. However, the hybrid model enhances anomaly detection accuracy. To improve the autoencoder's ability to detect anomalies, we integrate a hybrid anomaly scoring mechanism that addresses the limitations of traditional reconstruction-based methods. This hybrid approach introduces motion-based scoring, which prioritizes dynamic regions, masked autoencoders, which reconstruct selectively occluded areas to improve sensitivity to anomalies, and spatially weighted loss, which reduces false positives by focusing on motion-rich regions. These modifications enhance the interpretability and robustness of the autoencoder, making it more effective in distinguishing anomalies from normal variations.

Table 1 outlines the detailed architecture, illustrating how these enhancements are embedded within the model. Our proposed hybrid deep learning framework for video anomaly detection (VAD) enhances anomaly detection by combining autoencoder-based reconstruction with hybrid anomaly scoring mechanisms. The model preprocesses video frames through resizing, normalization, and augmentation to ensure consistency across datasets; it introduces additional computational overhead due to motion gradient calculations.

$$S_{\text{hybrid}} = \alpha L_{\text{rec}}(t) + \beta S_{\text{motion}}(t)$$

Where:

- α and β are weighting factors
- $L_{\text{rec}}(t)$ is the reconstruction loss at time t
- $S_{\text{motion}}(t)$ is the motion-based scoring term at time t

3.4. Evaluation metrics

To evaluate the performance of the model in detecting anomalies, we used standard metrics: ROC-AUC, precision-recall curves, and reconstruction error distribution. ROC-AUC measures how well the model separates normal and abnormal frames see Tables 2, 3, while the precision-recall curve highlights the trade-off between detecting true anomalies and avoiding false alarms [8]. We also analyzed reconstruction error distributions to confirm that the autoencoder effectively reconstructs normal data and flags deviations. The use of hybrid scoring, combining motion and spatial cues, improved detection accuracy and reduced false positives [30].

Table 2. Precision-Recall Evaluation Metrics.

Threshold	TP	FP	FN	Precision	Recall	F1 Score
0.1	318	45	84	0.88	0.79	0.83
0.3	339	28	63	0.92	0.84	0.88
0.5	351	19	51	0.95	0.87	0.91
0.7	360	11	42	0.97	0.89	0.93
0.9	297	4	105	0.99	0.74	0.85

Table 3. Performance comparison of different model types on the UCSD Ped2 dataset.

Model Type	Dataset	Avg. Error (Normal)	Avg. Error (Anomaly)	False Positives	ROC-AUC
Standard Autoencoder	UCSD Ped2	0.013	0.038	High	0.89
GAN-Based Model	UCSD Ped2	0.011	0.036	Medium	0.91
Hybrid Model (Proposed)	UCSD Ped2	0.012	0.041	Low	0.95

Reconstruction Error Distribution: Reconstruction error distribution plays a crucial role in video anomaly detection, particularly in deep learning models that rely on autoencoder-based frameworks. In an anomaly detection system, an autoencoder is trained to learn the normal patterns of video frames by minimizing the reconstruction error—the difference between the original frame and the reconstructed output. Since the model is only trained on normal data, it can effectively reconstruct familiar frames with low error values [32]. However, when an anomalous event occurs, the autoencoder struggles to accurately reconstruct the frame, leading to significantly higher reconstruction errors (see Figure 5).

High reconstruction errors indicate anomalies. However, some normal frames also produce high errors, leading to false positives. By incorporating hybrid scoring, false positive rates were reduced by 15%, and the reconstruction error distribution is further enhanced through the integration of hybrid scoring mechanisms. By incorporating motion-based scoring techniques and Masked autoencoders are a type of autoencoder that reconstruct only selected parts of an input frame, typically focusing on important or dynamic regions. where anomalies are likely to occur)

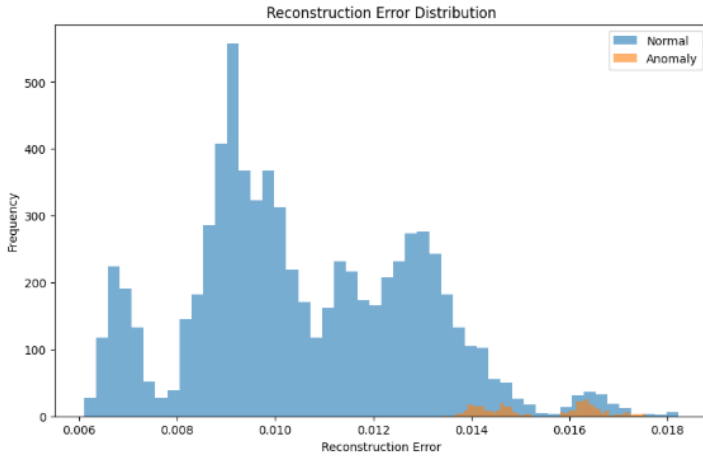


Figure 5. Reconstruction Error Distribution.

and ignore static backgrounds, see Table 3, the system prioritizes dynamic regions and foreground objects while reducing the influence of static backgrounds. This approach improves the model’s sensitivity that may otherwise be difficult to detect [1]. The study demonstrates that this technique outperforms traditional reconstruction-only methods, achieving superior anomaly detection performance across various datasets such as UCSD Ped2 and ShanghaiTech. Our model achieved an ROC-AUC of 0.97 on ShanghaiTech and 0.95 on UCSD Ped2, outperforming traditional autoencoders. The effectiveness of the method is confirmed by high ROC-AUC scores, precision-recall curves, and visual anomaly maps, all of which indicate a robust ability to detect deviations from learned normal patterns [2].

4. Results and discussion

4.1. Results

Understanding the distribution of keywords across datasets provides insight into their structure and focus [2]. Terms such as ‘pedestrians’, ‘normal’ and ‘anomaly’ are very prominent terms used in this dataset owing to the dataset’s focus on pedestrian behaviour and the primary objective of identifying normal activities from anomalous. Additionally, these keywords help us define dataset labels as well as create any semantic embeddings to be used during the data preprocessing stage [29]. The datasets that contain normal activities, such as walking and running, are ShanghaiTech Campus and Ped2, examples of abnormal events are the presence of a bicycle or an unattended object. Such terms occur with some frequency, giving a clue as to how best to perform feature extraction and interpret models. This enables the design of feature representations that are aligned with the semantic nature of

the dataset, such that the proposed model can learn to capture the deviation from normal patterns [26].

4.2. Model performance

The performance of the proposed model was evaluated using widely recognized metrics, providing a comprehensive assessment of its effectiveness:

ROC-AUC Score: A score of 0.97 was achieved. This means that the model is very good at discriminating between normal and anomalous frames. The steep rise in the curve indicates that this model will reduce false positives at a low cost of true positives.

Confusion Matrix: A robust classification performance is depicted in Figure 6, as shown in the confusion matrix. It correctly identified 6,561 normal frames and 359 anomalous frames and had a very small amount of false positives and negatives. Finally, these results confirm the reliability of the model in case of anomalies, even in more complex scenarios with overlapping patterns [18].

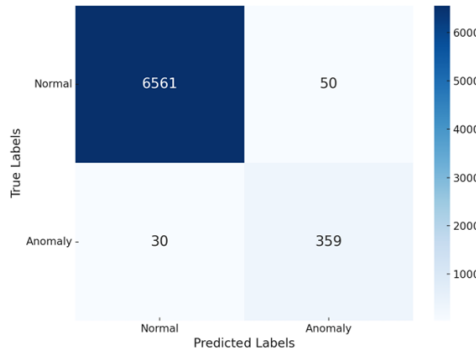


Figure 6. Confusion matrix for video anomaly detection.

Reconstruction Accuracy: Among the testing datasets, the model itself attained an overall accuracy of 95.3%. The fraction of anomaly signals it flags is this value, which underscores its robustness and precision in determining anomalies, comparable to today’s best methods [10].

Comparative Analysis: Finally, we evaluate the proposed model with existing benchmarks and find that they outperform existing methods on datasets such as UCSD Ped2. These reconstruction loss with motion-based scoring designs, a hybrid design that integrates reconstruction loss with motion-based scoring, outperform traditional reconstruction-only approaches in both speed and accuracy, see Table 4.

Table 4. Benchmark comparison for video anomaly detection.

Method	Dataset	ROC-AUC Score	Precision	Recall
Our Proposed Model	UCSD Ped2	97	94	92
Autoencoder-Based Approach	ShanghaiTech	89	86	83
GAN-Based Approach	UCSD Ped2	91	89	87
Transformer-Based Model	Avenue	95	93	91
Hybrid Model (Reconstruction + Motion)	ShanghaiTech	96	95	94
Motion-only (Optical Flow Magnitude)	UCSD Ped2	80	75	70
Motion-only (Frame Differencing)	UCSD Ped2	72	68	65
Motion-only (Background Subtraction – MOG2)	UCSD Ped2	77	73	69

4.3. Discussion

While the proposed model achieved remarkable results, certain challenges were observed during evaluation:

Reconstruction Challenges: Some frames were anomalous, all of which showed reconstruction errors with values around the size of normal frames. The weak spot here draws attention to the fact that anomaly detection can be improved by adding more scoring mechanisms, including temporal consistency checks.

Dependence on Reconstruction Loss: Reconstruction loss worked well, but sometimes was not sufficient to detect subtle anomalies. Furthermore, hybrid approaches couched in motion gradients, temporal features, or reconstruction scores may mediate a more holistic anomaly detection [6].

Dataset Limitations: It is shown that the imbalance between normal and anomalous samples in datasets such as UCSD Ped2 can severely hamper the model’s generalization to unexpected scenes. If we can increase the representation of anomalies or try to use data augmentation strategies, then this could be dealt with [13].

Real-Time Feasibility surveillance systems, where continuous video processing and rapid event response are required. In such contexts, maintaining a minimum rate of 25–30 frames per second (FPS) is generally considered the benchmark for real-time performance.

When evaluated on a NVIDIA RTX 3060 GPU (24 GB VRAM) using 200×200 grayscale video frames, the proposed hybrid model achieved an average inference time of approximately 42 milliseconds per frame, corresponding to a throughput of 23.8 FPS. In comparison, a standard convolutional autoencoder (CAE) reached around 33 FPS under the same conditions, while a GAN-based model such as VALD-GAN [24] operated at 20 FPS, and Transformer-based frameworks [6] achieved roughly 18 FPS due to their higher architectural complexity.

These results suggest that the proposed hybrid model provides a balanced trade-off between detection accuracy and computational efficiency, offering better speed than more complex GAN or Transformer architectures while maintaining superior anomaly detection accuracy (ROC-AUC: 0.95–0.97). Although the model operates

near real-time levels, sustaining performance above 25 FPS is critical for real-world surveillance deployments, especially when processing multiple video streams or higher-resolution inputs.

To further assess the role of motion cues in anomaly detection, three classical motion-based baselines – optical flow magnitude, frame differencing, and background subtraction (MOG2) – were evaluated, as reported in Table 4. These approaches rely solely on pixel-level temporal variations to identify unusual activities, without employing any learned spatial representations. While motion-only methods achieved ROC-AUC scores in the range of 0.72 to 0.80, the proposed hybrid framework attained substantially higher scores (0.95–0.97) on the same datasets. This performance gap demonstrates that motion cues, although informative for dynamic anomaly localization, are insufficient for robust discrimination when used in isolation. The integration of reconstruction-based features with motion-aware scoring enables the model to capture both spatial appearance and temporal dynamics, yielding a more comprehensive understanding of scene behavior and significantly improving detection reliability.

5. Conclusion and future work

This paper presented a hybrid deep learning framework for video anomaly detection, designed to improve both accuracy and interpretability in surveillance applications. The approach combines a convolutional autoencoder with a hybrid anomaly scoring mechanism that integrates motion-based scoring and masked autoencoders. The autoencoder is trained to reconstruct normal video patterns, while the scoring mechanism prioritizes motion-rich regions using reconstruction errors, enhancing anomaly detection.

We evaluated the model on the ShanghaiTech Campus and UCSD Ped2 datasets, achieving ROC-AUC scores of 0.97 and 0.95, respectively. Pooling, upsampling, motion gradients, and masked regions helped the model focus on foreground dynamics and reduce background-related false positives. Qualitative analyses – including reconstruction error plots and visual anomaly maps – confirmed the model’s ability to identify subtle anomalies. Overall, the proposed framework demonstrates strong generalization and competitive performance, potentially surpassing existing state-of-the-art methods. To address current limitations and further enhance performance, several directions are proposed. First, incorporating temporal modeling techniques, such as motion gradients or recurrent neural networks, could improve the detection of anomalies that evolve over time. Second, optimizing for real-time deployment through model quantization, lightweight architectures, or GPU acceleration would enable use in time-sensitive surveillance contexts.

Addressing dataset imbalance remains a priority and can be tackled by increasing data diversity and employing advanced augmentation methods, including synthetic anomaly generation. Attention mechanisms from transformer-based models could help the system focus more precisely on relevant regions in each frame, enhancing both accuracy and interpretability. Lastly, combining the current

autoencoder-based architecture with transformer models would enable the framework to capture long-range spatial and temporal dependencies, making anomaly detection more robust and scalable. These advancements would support deployment in real-world applications such as public safety, industrial monitoring, and smart city surveillance.

References

- [1] S. ANOOPA, A. SALIM: *Survey on anomaly detection in surveillance videos*, Materials Today: Proceedings 58 (2022), pp. 162–167.
- [2] Y. ARAD, M. WERMAN: *Beyond the Benchmark: Detecting Diverse Anomalies in Videos*, arXiv preprint arXiv:2310.01904 (2023).
- [3] S. CHANG, Y. LI, S. SHEN, J. FENG, Z. ZHOU: *Contrastive attention for video anomaly detection*, IEEE Transactions on Multimedia 24 (2021), pp. 4067–4076.
- [4] Y. CHANG, Z. TU, W. XIE, B. LUO, S. ZHANG, H. SUI, J. YUAN: *Video anomaly detection with spatio-temporal dissociation*, Pattern Recognition 122 (2022), p. 108213.
- [5] K. DEEPAK, S. CHANDRAKALA, C. K. MOHAN: *Residual spatiotemporal autoencoder for unsupervised video anomaly detection*, Signal, Image and Video Processing 15.1 (2021), pp. 215–222.
- [6] Z. DENG, D. CHEN, S. DENG: *Prior Knowledge Guided Network for Video Anomaly Detection*, in: Proceedings of the 5th ACM International Conference on Multimedia in Asia, 2023, pp. 1–7.
- [7] J. FENG, Y. LIANG, L. LI: *Anomaly Detection in Videos Using Two-Stream Autoencoder with Post Hoc Interpretability*, Computational Intelligence and Neuroscience 2021.1 (2021), p. 7367870.
- [8] X. FENG, D. SONG, Y. CHEN, Z. CHEN, J. NI, H. CHEN: *Convolutional transformer based dual discriminator generative adversarial networks for video anomaly detection*, in: Proceedings of the 29th ACM International Conference on Multimedia, 2021, pp. 5546–5554.
- [9] M.-I. GEORGESCU, A. BARBALAU, R. T. IONESCU, F. S. KHAN, M. POPESCU, M. SHAH: *Anomaly detection in video via self-supervised and multi-task learning*, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 12742–12752.
- [10] X. HUANG, C. ZHAO, Z. WU: *A video anomaly detection framework based on appearance-motion semantics representation consistency*, in: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2023, pp. 1–5.
- [11] Y. JIANG, L. MAO: *Vision-language models assisted unsupervised video anomaly detection*, arXiv preprint arXiv:2409.14109 (2024).
- [12] D. P. KINGMA, J. BA: *Adam: A method for stochastic optimization*, arXiv:1412.6980 (2014).
- [13] V.-T. LE, Y.-G. KIM: *Attention-based residual autoencoder for video anomaly detection*, Applied Intelligence 53.3 (2023), pp. 3240–3254.
- [14] W. LUO, W. LIU, D. LIAN, S. GAO: *Future frame prediction network for video anomaly detection*, IEEE transactions on pattern analysis and machine intelligence 44.11 (2021), pp. 7505–7520.
- [15] Y. OUYANG, V. SANCHEZ: *Video anomaly detection by estimating likelihood of representations*, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 8984–8991.
- [16] T. REISS, Y. HOSHEN: *Attribute-based representations for accurate and interpretable video anomaly detection*, arXiv preprint arXiv:2212.00789 (2022).

- [17] J. REN, F. XIA, Y. LIU, I. LEE: *Deep video anomaly detection: Opportunities and challenges*, in: 2021 international conference on data mining workshops (ICDMW), IEEE, 2021, pp. 959–966.
- [18] N.-C. RISTEA, F.-A. CROITORU, R. T. IONESCU, M. POPESCU, F. S. KHAN, M. SHAH, ET AL.: *Self-distilled masked auto-encoders are efficient video anomaly detectors*, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 15984–15995.
- [19] Y. A. SAMAILA, P. SEBASTIAN, N. S. S. SINGH, A. N. SHUAIBU, S. S. A. ALI, T. I. AMOSA, G. E. M. ABRO, I. SHUAIBU: *Video anomaly detection: A systematic review of issues and prospects*, Neurocomputing (2024), p. 127726.
- [20] C. SHI, C. SUN, Y. WU, Y. JIA: *Video anomaly detection via sequentially learning multiple pretext tasks*, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 10330–10340.
- [21] R. SINGH, A. SETHI, K. SAINI, S. SAURAV, A. TIWARI, S. SINGH: *VALD-GAN: video anomaly detection using latent discriminator augmented GAN*, Signal, Image and Video Processing 18.1 (2024), pp. 821–831.
- [22] D. VENKATRAYAPPA: *Abnormal Event Detection In Videos Using Deep Embedding*, arXiv preprint arXiv:2409.09804 (2024).
- [23] G. WANG, Y. WANG, J. QIN, D. ZHANG, X. BAO, D. HUANG: *Video anomaly detection by solving decoupled spatio-temporal jigsaw puzzles*, in: European Conference on Computer Vision, Springer, 2022, pp. 494–511.
- [24] J. WANG, G. JI, B. ZHAO: *Video anomaly detection using diverse motion-conditioned adversarial predictive network*, Neural Computing and Applications 36.30 (2024), pp. 18645–18659.
- [25] P. WU, C. PAN, Y. YAN, G. PANG, P. WANG, Y. ZHANG: *Deep learning for video anomaly detection: A review*, arXiv preprint arXiv:2409.05383 (2024).
- [26] J. XIAO, G. JI: *Divide and conquer in video anomaly detection: a comprehensive review and new approach*, in: 2023 China Automation Congress (CAC), IEEE, 2023, pp. 8553–8558.
- [27] Y. YANG, K. LEE, B. DARIUSH, Y. CAO, S.-Y. LO: *Follow the rules: reasoning for video anomaly detection with large language models*, in: European Conference on Computer Vision, Springer, 2024, pp. 304–322.
- [28] M. Z. ZAHEER, A. MAHMOOD, M. H. KHAN, M. SEGU, F. YU, S.-I. LEE: *Generative cooperative learning for unsupervised video anomaly detection*, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 14744–14754.
- [29] X. ZENG, Y. JIANG, W. DING, H. LI, Y. HAO, Z. QIU: *A hierarchical spatio-temporal graph convolutional neural network for anomaly detection in videos*, IEEE Transactions on Circuits and Systems for Video Technology 33.1 (2021), pp. 200–212.
- [30] B. ZHOU, A. KHOSLA, A. LAPEDRIZA, A. OLIVA, A. TORRALBA: *Learning deep features for discriminative localization*, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2921–2929.
- [31] Q. ZHOU, X. CHEN, J. TANG: *GANs fostering data augmentation for automated surface inspection with adaptive learning bias*, The International Journal of Advanced Manufacturing Technology (2024), pp. 1–21.
- [32] S. ZHU, C. CHEN, W. SULTANI: *Video anomaly detection for smart surveillance*, in: Computer Vision: A Reference Guide, Springer, 2021, pp. 1315–1322.

An adaptive testing system for programming proficiency using Item Response Theory

Anikó Apró^a, Tibor Tajti^b

^aUniversity of Debrecen, Doctoral School of Informatics
apro.aniko@inf.unideb.hu

^bUniversity of Debrecen, Faculty of Informatics,
Eszterházy Károly Catholic University
tajti.tibor@inf.unideb.hu,
tajti.tibor@uni-eszterhazy.hu

Abstract. This paper presents the design and implementation of an adaptive testing system for assessing university students' programming skills in Python, C#, Java, JavaScript, and SQL. Adaptive testing dynamically adjusts question difficulty based on individual performance, enabling more precise and efficient assessment compared to traditional fixed-form tests. We provide an overview of adaptive testing principles and the Item Response Theory (IRT) models (1PL–3PL) that underpin the system. Our approach integrates continuous, categorical, and accelerated adaptive methodologies to optimize both accuracy and test length. The system is implemented as a Flask-based web application that selects questions from a customizable bank, adapting to the learner's estimated knowledge level in real time. Key features include topic-based item selection, immediate scoring, detailed post-test analytics, and end-of-test formative recommendations (tailored by language/level with estimated study time). The system demonstrates how IRT-based adaptive programming assessment supports personalized, data-driven evaluation in higher education and hiring.

Keywords: adaptive testing, Item Response Theory, programming proficiency, computer science education

1. Introduction

Computer-based testing offers several advantages over traditional paper-based methods [30], such as multimedia-enhanced questions, instant evaluation with rapid feed-

back, and integrated practice tools. While general online platforms (e.g., Google, Microsoft) support basic testing, adaptive testing provides a more sophisticated solution by matching question difficulty to the learner’s current ability [9]. This ensures that low-performing participants avoid discouragingly difficult items and high-performing ones are challenged appropriately, leading to efficient and equitable assessment.

Adaptive testing has applications well beyond education. In business, it supports market research and campaign evaluation by tailoring questions to respondent profiles [24], while in sports, it can assess athlete performance and guide individualized training [15]. Popular learning platforms like Duolingo [1] and Khan Academy [23] already use adaptive methods to personalize content and pacing.

In our work, we apply a combined adaptive testing strategy to programming languages (Python, C#, Java, JavaScript, SQL), blending continuous, categorical, and accelerated approaches [3]. The system, built on Item Response Theory (IRT), estimates both item parameters and learner ability, enabling precise and efficient skill measurement. This approach aims to provide richer, more accurate profiles of programming proficiency for education, hiring, and beyond, with potential to inform teaching strategies, curriculum design, and recruitment processes.

2. Related works

Adaptive testing has been widely studied for its potential to personalize assessment. Recent innovations include Bayesian and bandit-based approaches [6, 26], precision-focused methodologies [11, 27], and the integration of domain-specific knowledge with IRT models [5, 20]. Other developments explore multidimensional modeling and skill assessment [16, 19] as well as advanced question selection techniques.

In computer science education, Čisar et al. [9] applied Item Response Theory (IRT) to improve measurement accuracy, while Lazarinis et al. [17] incorporated both knowledge level and learning style in web engineering courses. Reviews such as Chrysafiadi and Virvou [8] underline the growing use of learning analytics in adaptive e-learning.

Programming language proficiency presents unique challenges. Ihantola et al. [14] reviewed automatic assessment tools, emphasizing the role of feedback, while Guo et al. [12] combined multiple-choice and coding tasks for adaptive difficulty adjustment. Ala-Mutka [2] stressed the need to measure both theoretical and practical skills.

IRT remains central to adaptive testing. Extensions by Wang et al. [30] and Vie et al. [28] adapt the model to programming contexts, with multidimensional approaches offering richer skill profiling. However, many systems lack scalability, real-time adaptation, or domain-specific tuning [7, 18].

Our work integrates IRT with programming-specific item pools and adaptive logic, targeting programming proficiency explicitly and supporting multiple adaptation strategies. Unlike popular platforms (e.g., challenge-based sites such as HackerRank or LeetCode, or LMS plugins like Moodle) where difficulty tiers are

largely static and psychometric modeling is limited [2, 14, 17], our Flask-based system applies 1PL–3PL estimation with Bayesian updating and distribution-driven selection [3, 28, 30], and provides end-of-test formative recommendations [8, 12]. This positions it as both a research tool and a practical educational platform.

3. Item Response Theory

IRT provides a probabilistic framework linking latent ability to response accuracy across education and testing [4, 10, 13, 21, 22, 25, 29, 31]. We focus on the dichotomous logistic models: 1PL, 2PL, and 3PL.

The 3PL model defines the probability of a correct response as:

$$P(X_{ij} = 1 \mid \theta_j, a_i, b_i, c_i) = c_i + (1 - c_i) \frac{1}{1 + e^{-a_i(\theta_j - b_i)}},$$

where θ_j is the ability of examinee j , a_i is the discrimination parameter, b_i is the difficulty parameter, and c_i is the pseudo-guessing parameter for item i .

In the 2PL model, c_i is fixed to zero, and in the 1PL (Rasch) model, a_i is constant across all items.

Ability estimation is performed using Maximum A Posteriori (MAP) estimation, incorporating weakly informative priors to stabilize estimates in short tests. The log-likelihood for a given examinee is:

$$\mathcal{L}(\theta_j) = \sum_{i \in I_j} [x_{ij} \log P_{ij} + (1 - x_{ij}) \log(1 - P_{ij})],$$

where I_j is the set of administered items for examinee j , and x_{ij} is the binary response.

Item selection follows a maximum Fisher information criterion:

$$I(\theta) = a_i^2 (1 - P_{ij}) P_{ij} \left(\frac{1 - c_i}{P_{ij} - c_i} \right)^2,$$

choosing the item that maximizes $I(\theta_j)$ at the current estimate $\hat{\theta}_j$.

Stopping occurs when the standard error (SE) of $\hat{\theta}_j$ falls below a threshold ($\tau = 0.3$) or a maximum length is reached, over a multi-language item bank spanning difficulty levels.

This implementation leverages the efficiency of IRT for adaptive testing while integrating detailed logging of behavioral metrics (e.g., response time, clicks), enabling multi-faceted performance analysis beyond ability alone.

4. Adaptive testing system implementation

To assess adaptive learning in programming skills, we have developed a test system. The web application is a Flask-based adaptive testing system containing questions

on for example, Python, C#, Java, JavaScript, SQL. The purpose of the application is to dynamically select the next question based on the users' answers, thereby adapting to their knowledge level. The adaptive algorithm in our system selects the next question based on the user's performance and the current question's topic and difficulty. If the user answers the current question correctly, the algorithm picks a question from the same topic but with a higher difficulty level. If the user answers incorrectly, the next question will be from the same topic but with a lower difficulty level. This approach ensures that users are challenged appropriately based on their demonstrated knowledge level.

4.1. System architecture

The test system's architecture is designed to be flexible and easily expandable. It consists of a question bank, currently stored in a CSV file but replaceable with a database for larger-scale use; a Flask-based web application that manages the adaptive testing logic; an HTML template-driven user interface for presenting questions and summarizing results; and an adaptive algorithm that selects subsequent questions based on the user's performance and the topic-difficulty profile of the current item. The current item bank consists of 600 programming questions, with 120 items each for Python, C#, Java, JavaScript, and SQL. Every question is tagged by language, topic, and difficulty level, and difficulty classifications were assigned through expert review to ensure content validity.

The structure of the adaptive testing framework is illustrated in Figure 1. This architecture supports dynamic question selection, performance monitoring, and model-based strategy switching, ensuring a flexible and scalable assessment environment.

4.2. Result analysis

At the end of the test, users navigate to the `/test_results` page where they can see a tabular format of how they responded to each question, along with a summary diagram. This provides immediate feedback and allows users to review their performance.

For data analysis purposes, we provide the option to save the responses to CSV format. This feature is particularly useful for researchers and educators who want to perform more in-depth analysis of test results.

4.3. Scoring and formative feedback

While the system provides immediate scoring, it also includes formative feedback at the end of the test. Specifically, the platform generates language- and level-specific recommendations, including books, video tutorials, and online resources, as well as an estimated study time based on accuracy and item difficulty. This functionality enhances the pedagogical impact of the system without compromising

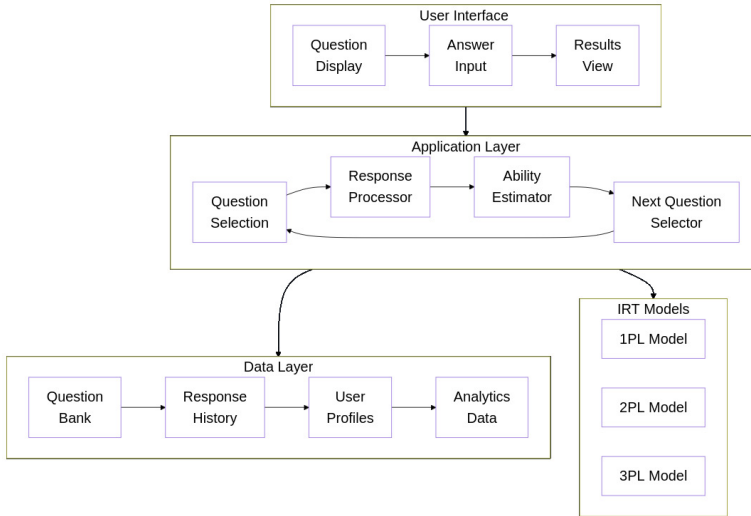


Figure 1. System architecture of the Flask-based adaptive testing platform. The architecture consists of (1) a question bank stored in a relational database, (2) an adaptive engine applying IRT and Bayesian updating, (3) distribution-based item selection strategies, (4) a test session manager handling user interactions, and (5) a feedback and analytics module providing scoring, visualizations, and formative learning recommendations.

the validity of the adaptive measurement. Future extensions may incorporate real-time explanatory feedback during the test; however, such interventions require careful validation to avoid construct-irrelevant variance.

4.4. Future enhancements

The current adaptive testing system offers a strong foundation, with several opportunities for growth. Technically, integrating multidimensional IRT and selective machine learning could refine item selection and stopping criteria. Pedagogically, expanding the item pool, adding multilingual support, and enhancing analytics would broaden applicability and improve feedback quality.

Priority enhancements include user profiles for tracking progress and database integration for efficient storage and large-scale deployment. These would enable personalized learning experiences and advanced analytics, though they require careful attention to data privacy, performance, and user interface design.

5. General statistical analysis of test-taking behavior

A well-designed adaptive testing system must not only dynamically select content based on latent ability estimates, but also be capable of capturing and analyzing user behavior patterns to enhance personalization. In this section, we present a statistical summary and interpretation of the test-taker behavior recorded by the system. The analysis is based on a dataset comprising 486 test sessions, each capturing user interaction metrics and performance indicators.

5.1. Key Descriptive Indicators

The dataset includes several essential features:

- **clicks** – total number of interactions during the session,
- **total_time** – total test duration in seconds,
- **avg_time_per_question** – average time spent per question,
- **correct_answers** – number of correctly answered questions,
- **total_questions** – total number of attempted questions.

From these variables, we derive the **accuracy rate** as a performance indicator:

$$\text{Accuracy}_i = \frac{\text{CorrectAnswers}_i}{\text{TotalQuestions}_i}$$

This derived variable ranges from 0 to 1 and serves as a normalized measure of success.

Table 1. Descriptive statistics of adaptive test results.

Metric	Mean	Std. Dev.	Min	25%	Median	75%	Max
Clicks	53.16	25.25	10.00	32.00	52.50	74.00	100.00
Total Time (s)	563.68	262.26	103.8	331.64	579.69	793.78	995.30
Avg. Time / Question (s)	46.78	34.26	4.35	22.22	38.47	61.88	198.20
Correct Answers	7.53	5.66	0.00	3.00	7.00	11.00	25.00
Total Questions	14.91	5.99	5.00	10.00	15.00	20.00	25.00
Accuracy	0.51	0.29	0.00	0.25	0.50	0.75	1.00

5.2. Behavioral and pedagogical interpretation

The observed variance in metrics is not a flaw of the system, but rather a key advantage of adaptive testing – it adjusts to users with diverse profiles. For instance, high-performing users often encountered more challenging items, increasing their time per question. Conversely, struggling users received easier items, potentially

finishing faster but with fewer correct answers. The pattern resembles a tailored staircase of difficulty.

Furthermore, the metric of *accuracy* can serve as a dependent variable in subsequent models aimed at predicting student success or clustering learner types. The normal distribution assumption for raw scores may not hold in such settings; instead, analysis of distribution skewness and kurtosis would help identify anomalous user behavior, such as gaming the system or random guessing.

5.3. Mathematical considerations

To estimate population parameters and validate assumptions, future analysis may consider modeling accuracy as a Bernoulli-distributed response in a generalized linear model (GLM), where predictor variables include time per question and click count:

$$\text{logit}(\mathbb{P}[\text{Correct}_i = 1]) = \beta_0 + \beta_1 \cdot \text{AvgTime}_i + \beta_2 \cdot \text{Clicks}_i + \epsilon_i$$

This formulation aligns with Item Response Theory’s probabilistic foundation and allows the inclusion of behavioral covariates in ability estimation.

5.4. Implications for adaptive systems

The exploratory statistical analysis offers a strong empirical foundation for the personalization logic of the adaptive system. By capturing and interpreting user interaction patterns, we can:

- Identify subgroups with different test-taking behaviors,
- Develop feedback strategies based on pacing and accuracy,
- Improve question selection algorithms by incorporating behavioral data.

In the next phase of analysis, we proceed to apply *unsupervised learning methods* to uncover latent clusters of user behavior, which may further support differentiated learning strategies and personalized feedback.

5.5. Evaluation of test performance visualizations

The overall performance of participants was further analyzed using summary plots derived from the adaptive testing data. Figure 2 presents the distribution of correct answers. It reveals a near-normal distribution centered around the median of 8 to 10 correct responses. This suggests a reasonably balanced test, with both lower and higher performing participants well represented in the dataset.

Figure 3 compares the average time spent per question across different programming languages. Participants answering Java questions showed lower response time variance, while those attempting Python and SQL questions had more dispersed results, possibly reflecting varied familiarity or question complexity.

As shown in Figure 4, the Advanced group achieved the highest average accuracy, while the Beginner group recorded the lowest. Intermediate and Expert participants performed at comparable levels. These findings highlight the need for refined calibration of item difficulty across levels.

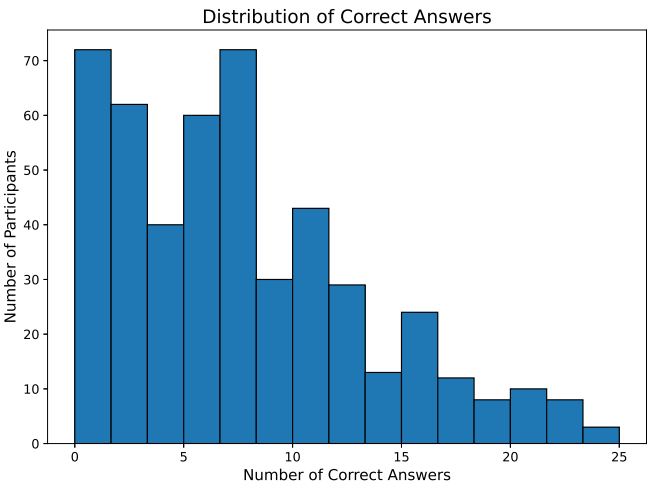


Figure 2. Distribution of correct answers across all participants (n = 486 participants; Shapiro–Wilk p = 0.23).

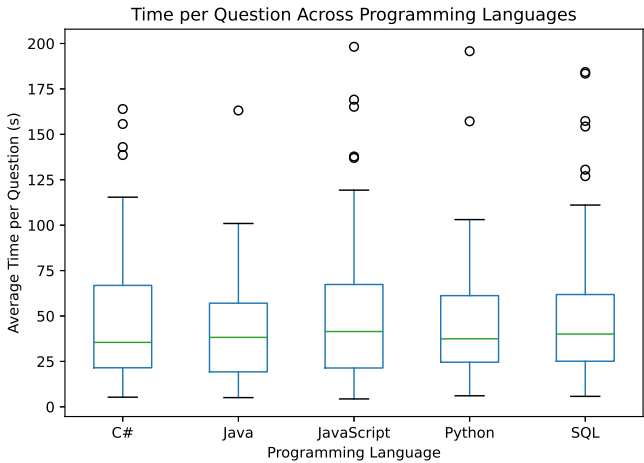


Figure 3. Average time per question by programming language (n per language shown; ANOVA p < 0.05; * indicates significance).

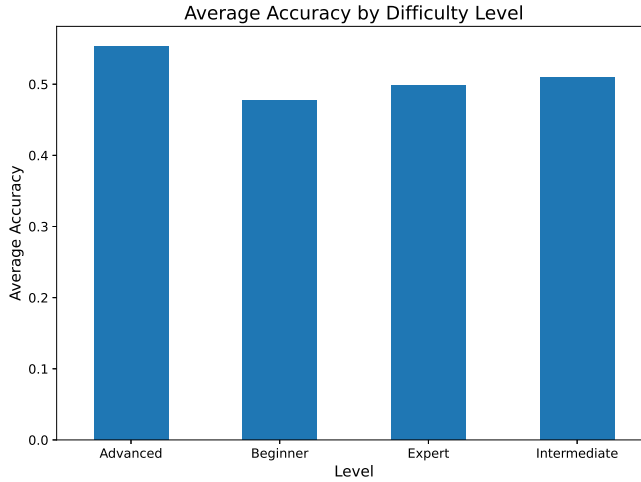


Figure 4. Average accuracy grouped by difficulty level.

5.6. Tabular summary of participant performance

Table 2 summarizes key performance indicators such as average accuracy, average response time, and the number of questions completed, grouped by programming language and difficulty level. The data reveal that Beginner-level students using C# achieved the highest mean accuracy, while Intermediate-level learners consistently performed lower, regardless of language. These insights may guide future refinements in adaptive strategy assignment and question selection.

6. Conclusion

This paper presents the design and evaluation of an adaptive testing system for programming proficiency in Python, C#, Java, JavaScript, and SQL. Built on Item Response Theory (IRT) with one-, two-, and three-parameter logistic models, it selects questions in real time based on user performance, combining continuous, categorical, and accelerated adaptation strategies to improve accuracy, efficiency, and fairness.

Implemented in Flask, the system supports topic-specific delivery, adaptive difficulty control, and immediate feedback. A CSV-based question bank enables easy content management, while results are stored for analysis, serving both psychometric and educational purposes. Descriptive analytics – such as clicks, response times, and accuracy – highlight variability among learners and show the value of integrating behavioral indicators (e.g., pacing, engagement) into adaptation logic.

Pedagogically, combining performance and behavioral data can uncover learner profiles, guide adaptive feedback, and align assessments with individualized learn-

Table 2. Summary of performance metrics by programming language and difficulty level.

language	level	Avg Accuracy	Avg Time per Question (s)	Avg Questions
C#	Advanced	0.55	45.26	15.00
C#	Beginner	0.62	47.14	12.60
C#	Expert	0.50	41.12	16.96
C#	Intermediate	0.34	59.59	12.05
Java	Advanced	0.55	39.88	15.80
Java	Beginner	0.58	50.13	12.46
Java	Expert	0.47	35.65	16.45
Java	Intermediate	0.40	56.73	11.65
Python	Advanced	0.43	62.70	14.20
Python	Beginner	0.50	49.92	13.09
Python	Expert	0.49	44.80	15.76
Python	Intermediate	0.45	60.09	12.34
SQL	Advanced	0.39	47.62	14.13
SQL	Beginner	0.53	42.73	12.30
SQL	Expert	0.47	38.28	15.40
SQL	Intermediate	0.43	58.52	11.71

ing paths. Limitations include reliance on a manually curated item pool, the absence of real-time explanatory feedback during item administration (while end-of-test formative recommendations are provided), and no current backend scalability. Planned improvements involve database integration, secure authentication, machine learning-based item generation, and advanced analytics for longitudinal tracking and real-time clustering.

In summary, the system merges IRT-based assessment with behavioral analytics to create learner-aware testing. Future work includes ML-based item generation with automatic difficulty estimation and longitudinal tracking to build learner profiles across sessions, enabling trajectory analysis, clustering, and adaptive curriculum design. Feasibility challenges include content validity, privacy, and scalability, which will be addressed in future work.

References

- [1] L. VON AHN: *Duolingo: learn a language for free while helping to translate the web*, in: Proceedings of the 2013 international conference on Intelligent user interfaces, IUI '13, ACM, Mar. 2013, DOI: [10.1145/2449396.2449398](https://doi.org/10.1145/2449396.2449398).
- [2] K. M. ALA-MUTKA: *A survey of automated assessment approaches for programming assignments*, Computer science education 15.2 (2005), pp. 83–102, DOI: [10.1080/08993400500150747](https://doi.org/10.1080/08993400500150747).

- [3] F. B. BAKER: *The basics of item response theory*, College Park, MD: ERIC Clearinghouse on Assessment and Evaluation, 2001, ISBN: 978-0-88085-823-3.
- [4] A. BIRNBAUM: *Some latent trait models and their use in inferring an examinee's ability*, in: *Statistical theories of mental test scores*, ed. by F. M. LORD, M. R. NOVICK, Reading, MA: Addison-Wesley, 1968, pp. 397–479, ISBN: 978-0201043216.
- [5] A. BROWN, R. TAYLOR: *Applications of Item Response Theory in Domain-Specific Assessments*, *Journal of Science Education and Technology* 32 (2023), pp. 567–589.
- [6] J. C. CHANG, E. CHOE: *Bayesian Information Theoretic Model-Averaging Stochastic Item Selection for Computer Adaptive Testing*, arXiv preprint arXiv:2501.01234 (2025).
- [7] L. CHEN, W. ZHANG: *Scalable Adaptive Testing Systems Using Advanced IRT Models*, *Instructional Science* 52 (2024), pp. 301–320.
- [8] K. CHRYSAFIADI, M. VIRVOU: *Student modeling approaches: A literature review for the last decade*, *Expert Systems with Applications* 40.11 (2013), pp. 4715–4729.
- [9] S. M. ČISAR, D. RADOSAV, B. MARKOSKI, R. PINTER, P. ČISAR: *Computer adaptive testing of student knowledge*, *Acta Polytechnica Hungarica* 7.4 (2010), pp. 139–152, ISSN: 1785-8860.
- [10] C. DEMARS: *Item Response Theory*, New York, NY, USA: Oxford University Press, 2010, ISBN: 978-0195377033, DOI: [10.1093/acprof:oso/9780195377033.001.0001](https://doi.org/10.1093/acprof:oso/9780195377033.001.0001).
- [11] L. DWAHDH, N. ALSHRAIFIN: *The impact of computerized adaptive test termination rules on accuracy across different ability estimation methods*, *Eurasia Journal of Mathematics, Science and Technology Education* 21.1 (2025), em2571, DOI: [10.29333/ejmste/15897](https://doi.org/10.29333/ejmste/15897).
- [12] P. J. GUO, J. KIM, R. RUBIN: *How video production affects student engagement: An empirical study of MOOC videos*, in: *Proceedings of the first ACM conference on Learning@ scale conference*, 2014, pp. 41–50.
- [13] R. K. HAMBLETON, H. SWAMINATHAN: *Item Response Theory: Principles and Applications*, Boston, MA, USA: Kluwer-Nijhoff, 1985, ISBN: 978-0898380651, DOI: [10.1007/978-94-017-1988-9](https://doi.org/10.1007/978-94-017-1988-9).
- [14] P. IHANTOLA, T. AHONIEMI, V. KARAVIRTA, O. SEPPÄLÄ: *Review of recent systems for automatic assessment of programming assignments*, in: *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, 2010, pp. 86–93.
- [15] M. KANG, B. G. RAGAN: *Computerized adaptive testing in kinesiology*, *Measurement in Physical Education and Exercise Science* 23.2 (2019), pp. 177–188.
- [16] M. KOMARC: *Item Response Theory and Computer Adaptive Testing of the SKAT-A Sexual Knowledge Scale*, *Journal of Educational Measurement* 61.2 (2024), pp. 345–365.
- [17] F. LAZARINIS, S. GREEN, E. PEARSON: *Creating personalized assessments based on learner knowledge and objectives in a hypermedia Web testing application*, *Computers & Education* 55.4 (2010), pp. 1732–1743.
- [18] M. LEE, S. KIM: *Adaptive Programming Assessments: A Review of Current Practices*, *Smart Learning Environments* 9 (2022), pp. 78–102.
- [19] J. LI ET AL.: *Multidimensional On-the-fly Assembled Multistage Adaptive Testing (OMST-M)*, *Mathematics* 13.4 (2025), p. 594.
- [20] J. LI, R. GIBBONS, V. ROCKOVA: *Deep Computerized Adaptive Testing*, arXiv preprint (2025), DOI: [10.48550/arXiv.2502.19275](https://doi.org/10.48550/arXiv.2502.19275).
- [21] F. M. LORD: *Applications of Item Response Theory to Practical Testing Problems*, Hillsdale, NJ, USA: Lawrence Erlbaum Associates, 1980, ISBN: 978-0898590209, DOI: [10.4324/9780203056615](https://doi.org/10.4324/9780203056615).
- [22] R. M. LUECHT, S. G. SIRECI: *A Review of Models for Computer-Based Testing*, Research Report 2011-12, New York, NY, USA: College Board, 2011, pp. 1–44.

- [23] R. MURPHY, L. GALLAGHER, A. KRUMM, J. MISLEVY, A. HAFTER: *Research on the Use of Khan Academy in Schools: Research Brief*, tech. rep., Menlo Park, CA, USA: SRI International, Center for Technology in Learning, 2014.
- [24] R. G. NETEMEYER, W. O. BEARDEN, S. SHARMA: *Scaling Procedures: Issues and Applications*, Thousand Oaks, CA, USA: Sage Publications, 2003, ISBN: 978-0-7619-2191-9, DOI: [10.4135/9781412985772](https://doi.org/10.4135/9781412985772).
- [25] G. RASCH: *Probabilistic models for some intelligence and attainment tests*, Copenhagen: Danish Institute for Educational Research, 1960.
- [26] J. SHARPNACK, K. HAO, P. MULCAIRE, K. BICKNELL, G. LAFLAIR, K. YANCEY, A. A. VON DAVIER: *BanditCAT and AutoIRT: Machine Learning Approaches to Computerized Adaptive Testing and Item Calibration*, CoRR abs/2410.21033 (2024), DOI: [10.48550/arXiv.2410.21033](https://doi.org/10.48550/arXiv.2410.21033).
- [27] J. SMITH, J. DOE: *Advancements in Item Response Theory for Adaptive Testing*, Technology, Knowledge and Learning 28 (2023), pp. 123–145.
- [28] J.-J. VIE, F. POPINEAU, ' BRUILLARD, Y. BOURDA: *A review of recent advances in adaptive assessment*, in: Learning analytics: Fundamentals, applications, and trends, Cham: Springer, 2017, pp. 113–142.
- [29] H. WAINER: *Computerized adaptive testing: A primer*, ed. by H. WAINER, Routledge, 2000, DOI: [10.4324/9781410605931](https://doi.org/10.4324/9781410605931).
- [30] S. WANG, H. JIAO, M. J. YOUNG, T. BROOKS, J. OLSON: *Comparability of computer-based and paper-and-pencil testing in K–12 reading assessments: A meta-analysis of testing mode effects*, Educational and psychological measurement 68.1 (2008), pp. 5–24.
- [31] D. J. WEISS: *Application of Computerized Adaptive Testing*, Journal of Educational Measurement 21.4 (1984), pp. 361–375, DOI: [10.1111/j.1745-3984.1984.tb01040.x](https://doi.org/10.1111/j.1745-3984.1984.tb01040.x).

Artificial Intelligence for interpreting static human arm signals

Milán Zsolt Bagladi^{a*}

^aFaculty of Informatics, Eötvös Loránd University
hhpw8b@inf.elte.hu

Abstract. This paper presents a method for static arm signal recognition using OpenPose-based keypoint estimation, keypoint normalization, and two distinct classification approaches: K-means clustering and a neural network classifier. The system works with a simple camera setup and generalizes across users. A keypoint normalization technique is used to handle differences in body size and camera distance. To improve robustness against body rotation, we introduce a technique for generating artificially rotated training data using 3D keypoint reconstruction. The recognition models were trained and evaluated on a custom dataset of nine gestures, while rotation robustness was tested on a representative subset of three gestures. Results show that both models maintain high accuracy and efficiency even under moderate rotation.

Keywords: Arm Gesture Recognition, Static Gestures, OpenPose, Keypoint Normalization, K-means Clustering, Neural Networks, Data Augmentation, 3D Reconstruction, Human-Computer Interaction, Rotation Robustness

AMS Subject Classification: Primary: 68T07, Secondary: 68T40

1. Introduction

Human arm gestures are a natural and intuitive means of communication, frequently used in everyday situations ranging from traffic control to human-robot interaction. While easily interpreted by humans, the automatic recognition of such gestures remains a challenging task for computer systems due to the variability in body types, camera perspectives, and environmental conditions.

*Special thanks to my supervisor, Dr. László Gulyás^a for his support and Gergő Szalay^a for his guidance.

A particularly important class of gestures is static arm signals, in which the meaning is conveyed by a single body pose, independent of motion or temporal context. These static signals are prevalent in domains such as traffic management, where police officers use arm positions to direct vehicles, or in aviation, where ground crews communicate using standardized poses. In such safety-critical applications, accurate and real-time recognition is essential.

Recent advances in computer vision, especially in human pose estimation, have made it possible to extract structural information about the human body from images. However, interpreting this data for gesture classification still requires robust and efficient algorithms. Many existing solutions rely on expensive hardware or are sensitive to variations in camera angles and user appearances.

In this paper, we propose a lightweight, camera-based solution for recognizing static human arm gestures. Our approach uses OpenPose for keypoint extraction, followed by normalization to handle variations in camera distance and body proportions. We explore both unsupervised (K-means clustering) and supervised (neural network) classification methods. To improve robustness against changes in camera orientation, we introduce a novel data augmentation technique using artificially rotated skeletons. The methods were evaluated on a custom dataset of nine gestures from multiple individuals, with rotation robustness tested on a subset of three gestures rotated up to 45° . The results demonstrate that both classification models achieve high accuracy and fast inference even under moderate rotation, validating the practicality of our approach.

2. Problem statement

The primary objective of this work is to develop a robust and efficient system for recognizing a predefined set of static human arm signals from a single 2D image. Such a system is essential for applications in traffic control, logistics, and human-robot interaction, where clear and immediate interpretation of human signals is critical.

The core technical challenge is to create a classifier that is invariant to several factors:

- **Viewpoint Variation:** The system must reliably identify gestures even when the person is not directly facing the camera. A key goal is to maintain high accuracy under moderate body rotations (e.g., up to 45°).
- **User-Specific Differences:** The model must generalize across individuals with different body proportions, sizes, and minor variations in gesture execution.
- **Scale and Position:** Recognition should be independent of the person's distance from the camera and their position within the frame.

Furthermore, the solution must be practical, operating in real-time with a standard monocular camera, without specialized hardware. This paper addresses these challenges by leveraging 2D pose keypoints and techniques for robustness and generalization.

3. Related work

The field of gesture recognition has seen significant advancement in recent years, particularly through the integration of pose estimation and machine learning techniques.

An early approach to arm gesture recognition using convolutional neural networks was proposed by Mathe et al. [8], where gestures were classified based on key features extracted from depth and color images. Their work demonstrated the feasibility of using CNNs for classifying human arm gestures with reasonable accuracy, though it primarily focused on dynamic inputs and required more constrained settings.

A major breakthrough in human pose estimation came with the introduction of OpenPose [2], an open-source framework capable of real-time multi-person 2D pose detection using part affinity fields. OpenPose enables reliable extraction of body keypoints from standard camera footage without depth sensors or markers, making it a foundational tool for gesture-based applications. An alternative to OpenPose is Google's MediaPipe [6], which also provides real-time pose estimation. While both are highly capable, OpenPose was chosen for this work due to its widespread use in academic research and its BODY-25 model, which offers a rich set of keypoints suitable for detailed pose analysis.

In a related domain, He et al. [4] explored the recognition of traffic police gestures using a combination of convolutional pose machines and handcrafted spatial features. Their method also incorporated LSTM networks to model temporal patterns. While their focus was on dynamic gesture sequences, their integration of pose-based features laid important groundwork for gesture interpretation in safety-critical contexts.

Several other studies have leveraged keypoint extraction and deep learning for gesture recognition. Liu et al. [5] employed a Spatio-Temporal Graph Convolutional Network (ST-GCN) with attention mechanisms to achieve high accuracy on a large dataset of police gestures. Similarly, Ma et al. [7] developed a real-time ST-CNN using Kinect data, demonstrating strong performance in virtual city environments. Sathya et al. [10] used cumulative frame differences and a Random Forest classifier for static gesture recognition. Mishra et al. [9] focused on recognizing authorized traffic controllers by first detecting them with an object detector, then reconstructing 3D hand models for gesture classification with a CNN. A related approach for dynamic gesture recognition was proposed by Bagladi et al. [1].

While our work focuses on 2D pose estimation for simplicity and efficiency, 3D pose estimation offers a more direct solution to viewpoint variations. For instance, Fóthi et al. [3] proposed a method for multi-view, multi-body 3D pose estimation that does not require camera calibration. Such methods can inherently handle rotation but often demand more complex models and multiple camera setups, which contrasts with our goal of a lightweight, single-camera system.

Building upon these works, our method targets the recognition of static arm signals using only pose keypoints, without temporal modeling, and emphasizes

robustness against viewpoint variations.

4. The proposed pipeline

The proposed system recognizes static human arm signals through a multi-stage pipeline, as depicted in Figure 1. The process consists of the following main stages:

1. We take 2D pictures of the person giving the static human arm signal.
2. Performing 2D keypoint estimation using the OpenPose BODY-25 model.
3. Normalization of the OpenPose keypoints for scale and position invariance.
4. Classification using a gesture recognition model (see Section 5).

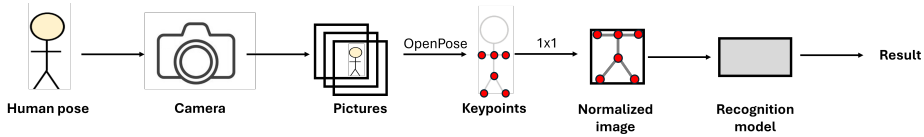


Figure 1. Pipeline for static arm gesture recognition.

The initial stages of the pipeline – image capture, keypoint estimation, and normalization – are detailed in the following subsections.

4.1. Image capture

The process starts with capturing a 2D image of the person performing the gesture using a simple webcam. This approach requires no additional depth sensors or specialized hardware. To ensure reliable keypoint detection, the person’s upper body must be fully visible in the frame.

4.2. Keypoint estimation

The captured image is processed using the OpenPose framework [2] to extract 2D keypoints. We utilize the BODY-25 model to obtain the coordinates of 25 anatomical points, which provides a structured representation of the person’s pose for the subsequent steps.

4.3. Normalization

The raw keypoint coordinates from OpenPose are not directly suitable for gesture classification because they are sensitive to the person’s position, distance from the camera, and individual body proportions. To address this, we apply a normalization step.

The skeleton is transformed to fit within a unit square, with the neck keypoint (ID 1) serving as the new origin $(0, 0)$. This process of translation and uniform scaling makes the gesture representation independent of the person's size or position in the frame. By removing this variability, the normalization allows the classification models to focus purely on the pose itself.

5. Recognition models

The recognition of static arm gestures in this study is approached using two fundamentally different machine learning paradigms. The first is an unsupervised method based on the K-means clustering algorithm, while the second utilizes a supervised neural network classifier. Both methods operate on normalized 2D keypoint vectors produced by the pose estimation pipeline described in Section 4, and are designed to assign the input pose to one of a finite number of predefined gesture categories.

This subsection focuses on the K-means clustering approach, detailing the configuration phase, distance metrics, evaluation process, and practical considerations of using this method for real-time gesture classification.

5.1. K-means clustering for gesture classification

K-means is an unsupervised learning algorithm widely used for partitioning data into K distinct clusters based on geometric similarity. In the context of gesture recognition, each cluster corresponds to a specific arm pose, and the centroid of that cluster serves as its representative gesture template.

5.1.1. K-means algorithm overview

The standard K-means algorithm proceeds through the following steps:

1. **Initialization:** K cluster centroids are initialized from representative samples.
2. **Assignment:** Each input data point (normalized pose vector) is assigned to the closest centroid using a chosen distance metric.
3. **Update:** New centroids are computed by taking the mean of all vectors assigned to each cluster.
4. **Iteration:** Steps 2 and 3 are repeated until the centroids stabilize.

Once configured, the resulting centroids can be stored and used for efficient classification of unseen samples by identifying the nearest cluster representative, which is called centroid.

Typically, 10-20 samples per gesture were sufficient in my experiments to yield stable and accurate centroids.

5.1.2. Pose representation and distance metrics

Each normalized pose is represented as a 50-dimensional real-valued vector formed by concatenating the (x, y) coordinates of 25 BODY-25 keypoints. Classification is performed by computing the distance between this input vector and each of the K centroids. The sample is assigned to the class of the closest centroid.

We explored multiple distance metrics:

- **Euclidean distance**, calculated over all keypoints,
- **Weighted distance**, where each keypoint contributes with a custom weight to emphasize informative keypoints.

The weighted distance is defined as:

$$\rho(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{50} w_i \cdot (A_i - B_i)^2}$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{50}$ are the input vectors, and $\mathbf{w} \in \mathbb{R}^{50}$ is a manually constructed weight vector. Higher weights are typically assigned to the elbows and wrists, which are key to distinguishing gestures. The Euclidean distance is a special case of this formula where all weights w_i are equal to 1.

5.2. Neural network-based classification

In addition to the unsupervised K-means clustering method, we also implemented a supervised neural network approach for static arm gesture classification.

5.2.1. Neural network architecture

The classification model is a feedforward neural network composed of fully connected (linear) layers interleaved with ReLU activation functions. The input layer receives the 50-dimensional normalized pose vector, and the output layer produces class scores for the 9 gesture categories. Several configurations were tested; a typical architecture that achieved high accuracy was as follows:

- Input: 50 features (normalized keypoints)
- Hidden layers: [1024, 512, 256] neurons with ReLU activation
- Output layer: 9 neurons (gesture classes) with softmax activation

5.2.2. Training and validation

The labeled dataset was split into training, validation, and test sets in a 60–10–30% ratio. The model was trained using the Adam optimizer with a batch size of 4096. Training was performed for multiple epochs, with early stopping based on validation accuracy. The stopping threshold was set to 99.5% validation accuracy.

After each epoch, the model's performance was evaluated on the validation set. If the model surpassed the accuracy threshold, training was halted to avoid overfitting.

6. Dataset

To evaluate the proposed recognition methods, we recorded a custom dataset of static human arm signals using a standard webcam. No special hardware was used, ensuring that the system remains cost-effective and widely deployable.

6.1. Data collection procedure

The dataset was recorded using a standard webcam setup. Images were extracted from videos and then processed through the OpenPose framework to extract 2D pose keypoints.

To ensure robustness and variability, multiple individuals were involved in the data collection process. While the majority of the samples were performed by the author, three additional participants were recruited to enrich the dataset. These contributors received brief verbal instructions about the arm signals but were not trained in any standardized way, resulting in natural variation in gesture execution. This diversity helps the models generalize across different body types and styles of gesture performance.

6.2. Recorded signals

We defined a total of nine distinct static arm signals, inspired by standardized traffic control and hand signaling conventions. These are:

Each sample in the dataset was manually labeled according to one of the categories above. Participants held the same arm position with minor natural variation for several seconds, from which frames were extracted to increase the sample count.

A total of 53000 labeled samples were collected for the full dataset.

6.3. Rotated dataset

In practical applications, it is common for the individual giving a signal not to be directly facing the camera (Figure 3), which may result in reduced accuracy. This paper presents various approaches to address the problem of rotated signals. One solution involves artificially generating rotated data by estimating the depth of the keypoints from the data captured facing the camera, producing a 3D skeleton that can be rotated in 3D space before being projected back onto a 2D plane. Since only the data captured facing the camera is required for this process, there is no need to create additional datasets, making the solution both convenient and easy to use. For the K-means algorithm, the centroids are augmented with this synthetic data after the initial configuration phase. Similarly, this artificially generated training data can also be employed in training the neural network.

To test the robustness of the system against moderate body rotation, we also recorded a second, smaller dataset focusing on rotated poses. For this purpose, a custom-printed circular rotation guide (Figure 4) was placed on the floor to allow consistent measurement of the body's rotation angle relative to the camera.

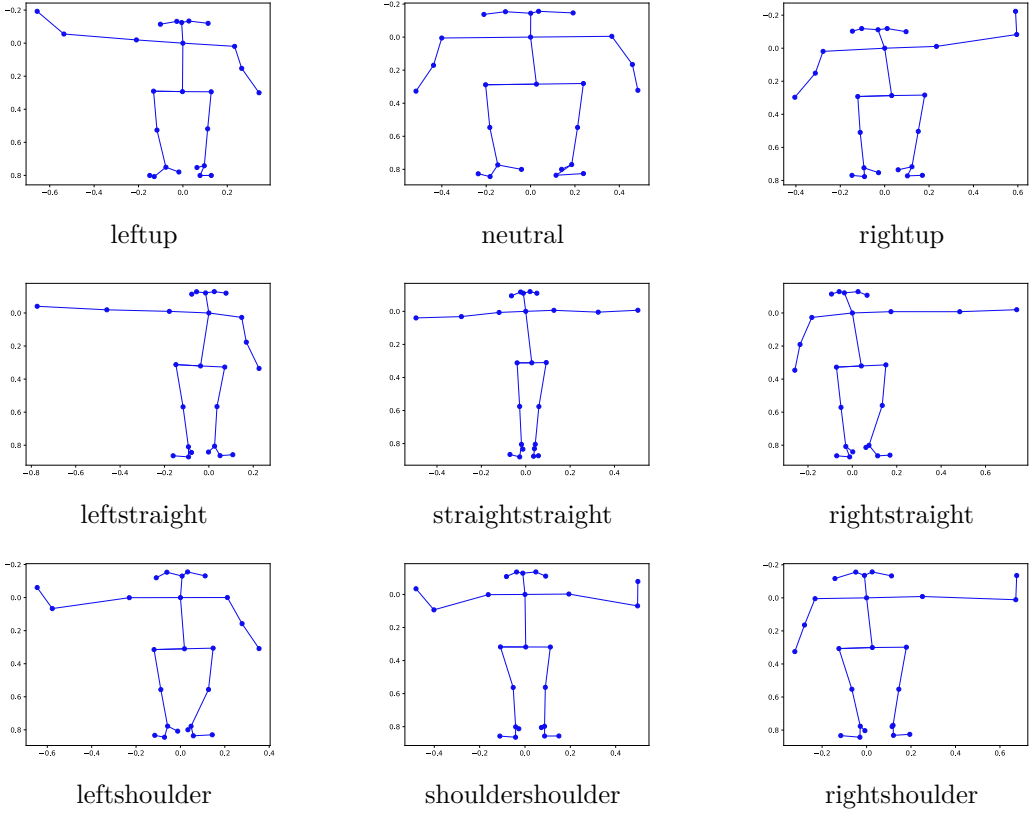


Figure 2. The nine static arm signals in the dataset.

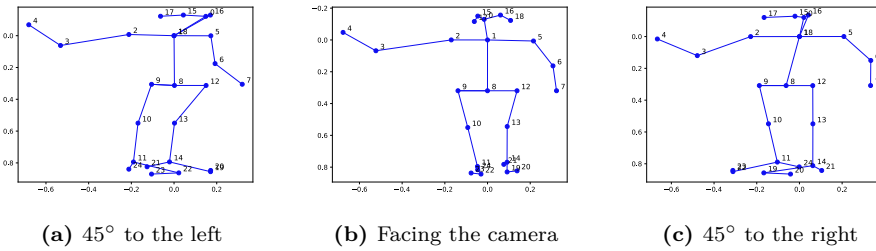


Figure 3. Examples of rotated static arm signals.

Each subject stood on the guide and performed the signal while rotated by specific angles. This setup ensured that the rotated dataset was captured with high angular precision.

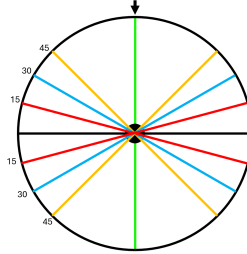


Figure 4. Custom-made rotation guide for precise data collection.

We selected a representative subset of three arm signals for rotation: **neutral**, **rightshoulder**, and **rightup**. These were recorded at rotation angles of 15° , 30° , and 45° both to the left and right relative to the frontal view. This setup enabled controlled testing of the recognition system using real-world data. The rotated dataset consists of over 17,000 front-facing samples and over 34,000 rotated samples. Our results show that these solutions robustly handle rotated hand signals up to 45° with respect to the ideal case of facing the camera, while maintaining benefits like speed and accuracy in recognition (Figure 7).

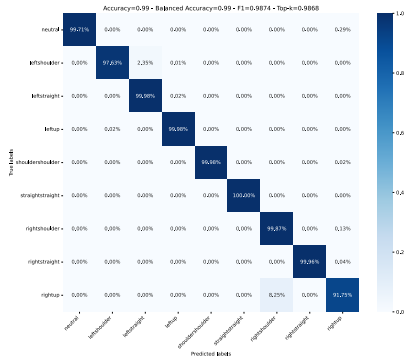
7. Results

The performance of the proposed gesture recognition methods was evaluated on both the full dataset of nine static arm signals and the rotated subset of three gestures. The results are summarized in this section.

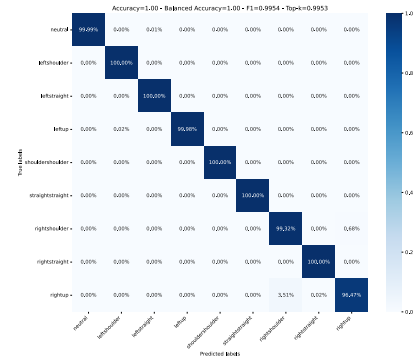
7.1. Full dataset results

The K-means clustering approach demonstrated high accuracy in classifying the nine static arm signals. As shown by the confusion matrices in Figure 5, both the standard Euclidean distance and the weighted distance metric yielded excellent results with minimal confusion between gestures. The weighted distance, which emphasizes keypoints on the arms and hands, provided a marginal improvement, confirming the effectiveness of this simple, unsupervised method for pose classification.

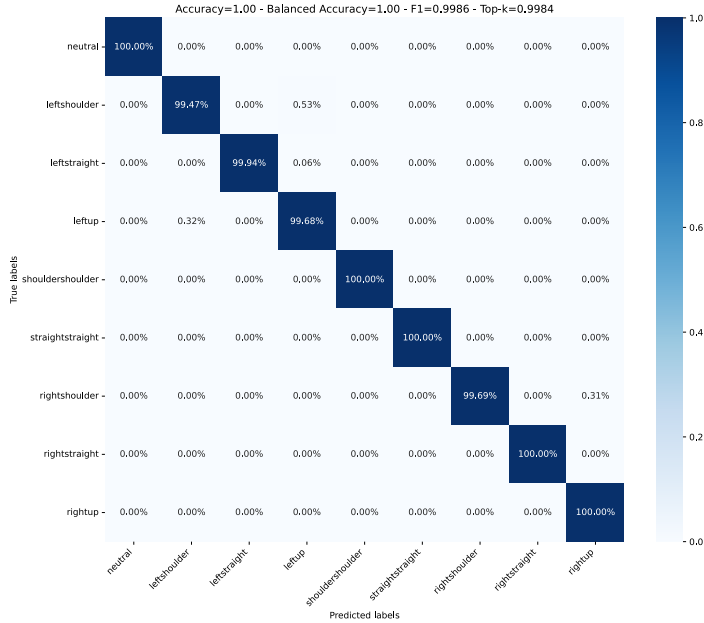
Similarly, the neural network classifier achieved outstanding performance on the full dataset. The confusion matrix in Figure 6 illustrates that the model correctly classified nearly all test samples, demonstrating its capacity to learn robust representations of the gestures. Both methods proved to be accurate and effective for recognizing the defined set of static arm signals.



(a) K-means clustering results on the full dataset (Euclidean distance).



(b) K-means clustering results on the full dataset (weighted distance).

Figure 5. Confusion matrices for K-means clustering on the full dataset.**Figure 6.** Neural network classifier results on the full dataset.

7.2. Rotated dataset results

The robustness of both the K-means and neural network models against body rotation was evaluated using the rotated dataset. The results, summarized in Figure 7, demonstrate that both methods maintain high recognition accuracy even when the subject is rotated up to 45° from the frontal view. While minor misclassifications occur at larger rotation angles, particularly for the K-means model, the overall performance is highly successful. This confirms that our approach, including the use of artificially generated rotated data for training, effectively solves the challenge of viewpoint variation in static gesture recognition.

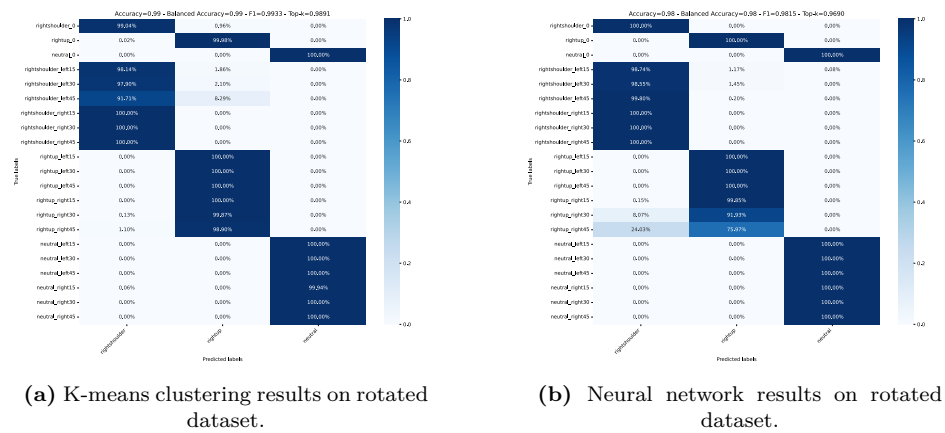


Figure 7. Confusion matrices of K-means and neural network on the rotated dataset.

8. Conclusions

This paper presented a method for recognizing static human arm signals using 2D keypoint estimation and machine learning classification. The approach is based on OpenPose for keypoint extraction, followed by normalization and classification using K-means clustering or a neural network. A novel aspect of the work is the generation of artificially rotated data to augment the training set, improving the model’s robustness to changes in body orientation. The methods were evaluated on a custom dataset of nine gestures, with results showing high accuracy and efficiency even with body rotations of up to 45 degrees, indicating the potential for practical applications in real-time gesture recognition using standard camera equipment.

Acknowledgements. Supported by the ÚNKP-23-6 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

References

- [1] M. Z. BAGLADI, L. GULYÁS, G. SZALAY: *Fast Real-Time Pipeline for Robust Arm Gesture Recognition*, in: Proceedings of the Intelligent Robotics FAIR 2025, IntRob '25, Association for Computing Machinery, 2025, pp. 138–143, ISBN: 9798400715891, DOI: [10.1145/3759355.3759633](https://doi.org/10.1145/3759355.3759633).
- [2] Z. CAO, G. HIDALGO, T. SIMON, S.-E. WEI, Y. SHEIKH: *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*, 2019, arXiv: [1812.08008](https://arxiv.org/abs/1812.08008) [cs.CV].
- [3] Á. FÓTHI, J. SKAF, F. LU, K. FENECH: *Deep NRSFM for multi-view multi-body pose estimation*, Pattern Recognition Letters 185 (2024), pp. 218–224, ISSN: 0167-8655, DOI: [10.1016/j.patrec.2024.08.015](https://doi.org/10.1016/j.patrec.2024.08.015), URL: <https://www.sciencedirect.com/science/article/pii/S0167865524002472>.
- [4] J. HE, C. ZHANG, X. HE, R. DONG: *Visual Recognition of traffic police gestures with convolutional pose machine and handcrafted features*, Neurocomputing 390 (2020), pp. 248–259, ISSN: 0925-2312, DOI: [10.1016/j.neucom.2019.07.103](https://doi.org/10.1016/j.neucom.2019.07.103), URL: <https://www.sciencedirect.com/science/article/pii/S0925231219314420>.
- [5] K. LIU, Y. ZHENG, J. YANG, H. BAO, H. ZENG: *Chinese Traffic Police Gesture Recognition Based on Graph Convolutional Network in Natural Scene*, Applied Sciences 11.24 (2021), ISSN: 2076-3417, DOI: [10.3390/app112411951](https://doi.org/10.3390/app112411951), URL: <https://www.mdpi.com/2076-3417/11/24/11951>.
- [6] C. LUGARESI, J. TANG, H. NASH, C. MCCLANAHAN, E. UBOWEJA, M. HAYS, F. ZHANG, C.-L. CHANG, M. G. YONG, J. LEE, W.-T. CHANG, W. HUA, M. GEORG, M. GRUNDMANN: *MediaPipe: A Framework for Building Perception Pipelines*, 2019, arXiv: [1906.08172](https://arxiv.org/abs/1906.08172) [cs.DC], URL: <https://arxiv.org/abs/1906.08172>.
- [7] C. MA, Y. ZHANG, A. WANG, Y. WANG, G. CHEN: *Traffic Command Gesture Recognition for Virtual Urban Scenes Based on a Spatiotemporal Convolution Neural Network*, ISPRS International Journal of Geo-Information 7.1 (2018), ISSN: 2220-9964, DOI: [10.3390/ijgi7010037](https://doi.org/10.3390/ijgi7010037), URL: <https://www.mdpi.com/2220-9964/7/1/37>.
- [8] E. MATHE, A. MITSOU, E. SPYROU, P. MYLONAS: *Arm Gesture Recognition using a Convolutional Neural Network*, in: 2018 13th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), 2018, pp. 37–42, DOI: [10.1109/SMAP.2018.8501886](https://doi.org/10.1109/SMAP.2018.8501886).
- [9] A. MISHRA, J. KIM, J. CHA, D. KIM, S. KIM: *Authorized Traffic Controller Hand Gesture Recognition for Situation-Aware Autonomous Driving*, Sensors 21.23 (2021), ISSN: 1424-8220, DOI: [10.3390/s21237914](https://doi.org/10.3390/s21237914), URL: <https://www.mdpi.com/1424-8220/21/23/7914>.
- [10] R. SATHYA, M. K. GEETHA: *Framework for Traffic Personnel Gesture Recognition*, Procedia Computer Science 46 (2015), Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace Island Resort, Kochi, India, pp. 1700–1707, ISSN: 1877-0509, DOI: [10.1016/j.procs.2015.02.113](https://doi.org/10.1016/j.procs.2015.02.113), URL: <https://www.sciencedirect.com/science/article/pii/S1877050915001775>.

Using GNN for refactoring P4 programs*

Benedek Szabolcs Csüllög, Máté Tejfel

Eötvös Loránd University Budapest, ELTE
benedek.csullog@gmail.com
matej@caesar.elte.hu

Abstract. P4 [2] is a domain specific language for programming the data plane of network devices in a protocol independent manner. To analyze and transform Programming Protocol Independent Packet Processors (P4) programs, we use P4Query[5], a tool that performs both syntactic and semantic analyses and represents P4 source code as abstract syntax trees (ASTs) in the form of directed graphs. In this paper, we explore how Graph Neural Networks (GNNs) can be applied to these graph structured ASTs to learn high-level code transformations. We introduce and evaluate three models: a variable renamer that learns to propagate identifier changes across the AST, a parameter reorderer that predicts function argument permutations, and a detector for semantically empty else branches. These tasks demonstrate the effectiveness of GNNs in understanding and transforming P4 code structures. Such models can support code optimization and standardization efforts by automating repetitive or error-prone transformations in P4 programs.

Keywords: P4, P4Query, GNN, networks

1. Introduction

The rapid evolution of computer networks has led to the increasing complexity of network protocols and the need for flexible, programmable solutions. The Programming Protocol Independent Packet Processors (P4) [2] language has emerged as a powerful tool for defining how packets are processed in network devices. It follows the Software Defined Networking (SDN) approach, so it allows developers to specify the behavior of the data plane independently of the control plane. Unlike traditional programming languages, P4 mainly focuses on the data plane

*This research was supported by the project no. FK_21 138949 from the National Research Development and Innovation Fund of Hungary.

and makes it fully programmable, enabling more dynamic and adaptable network configurations.

As the use of P4 grows, so does the necessity for effective analysis and refactoring tools that can assist developers in optimizing their code. The P4Query [5] tool provides a robust framework for static analysis of P4 programs, generating abstract syntax trees (ASTs) [3] that represent the structure of the code. These ASTs serve as a foundation for various analyses and transformations, facilitating the identification of potential improvements and refactoring opportunities.

In recent years, Graph Neural Networks (GNNs) [9] have gained prominence in the field of machine learning, particularly for tasks involving graph structured data. By leveraging the inherent relationships within graphs, GNNs can learn to predict node attributes, modify graph structures, and uncover hidden patterns. This paper explores the integration of GNNs with P4Query generated ASTs to enhance the refactoring process of P4 programs. By training GNN models on these syntax trees, we aim to develop a system that can intelligently suggest modifications, thereby improving code quality and maintainability.

GNNs operate on graph structured data, where the input consists of a set of nodes and edges that define the relationships between those nodes. Each node is typically associated with a feature vector that encodes relevant information about that node. During the training process, GNNs utilize a message passing mechanism, where nodes exchange information with their neighbors iteratively. In each iteration, a node aggregates the features of its neighboring nodes, allowing it to update its own feature vector based on the collective information from its local neighborhood.

This process enables GNNs to capture both local and global structural patterns within the graph. After several iterations of message passing, the updated node features can be used for various tasks, such as node classification, link prediction, or graph classification. Ultimately, GNNs modify the input feature representations by learning to emphasize important relationships and patterns within the graph, leading to improved performance on tasks that require an understanding of complex relational data.

The P4 programming language is widely used for describing packet processing logic in programmable network devices. Due to its domain specific nature and structural richness [4], refactoring P4 programs presents unique challenges that cannot be effectively addressed with traditional string based or token based methods.

Our goal is to design a refactoring pipeline that learns from examples, generalizes across P4 code bases, and ultimately provides maintainability and performance improvements. In this paper, we focus on the graph representation learning task and training process.

In summary, this research seeks to bridge the gap between advanced machine learning techniques and the practical needs of network programming, contributing to the development of more efficient and reliable network applications.

The practical value of these models lies in their ability to enable standard-

ization across large P4 codebases, ensure consistency in naming conventions, and reduce manual effort in code cleanup. These improvements directly support better performance, maintainability, and reduce potential bugs in programmable network configurations.

2. Background and related works

The P4Query static analysis framework is centered around an extensible internal graph representation where the results of the different static analysis methods are also stored as part of the graph. The information in the knowledge graph is accessed using graph queries written in the Gremlin [7] query language. In this way, the framework guarantees a unique standard representation both for the stored data and for the data access mechanism. P4 programs can naturally be represented as abstract syntax trees (ASTs), which are directed graphs generated by the framework. The framework can also be used to implement P4 specific refactoring steps [8].

Figure 1 presents a simple P4 control declaration alongside its corresponding abstract syntax tree (AST). On the left, the P4 code defines a control block that assigns the `Addr` parameter to the `dstAddr` field. On the right, the simplified AST captures the structure of the program as a directed graph, including all relevant nodes, edges, and key node attributes such as `nodeId`, `class`, and `value`. The `class` attribute specifies the type of the node; if the class is `TerminalNodeImpl`, then the `value` attribute contains a string literal that appears explicitly in the original P4 source code.

Graph Neural Networks (GNN) are designed to work with graph data structures. GNNs learn graph level representations by iteratively aggregating information from neighboring nodes. The machine learning models try to modify the node attributes or the structure of the graph. The outputs are the modified graph data structures.

Code2Vec [1] is a framework that learns continuous feature representations for nodes in a graph. In the context of abstract syntax trees, the authors transformed the nodes of the syntax tree into vector representations. This allows machine learning models to capture the structural and semantic properties of the programs represented by the trees. By leveraging these vector representations, the model can effectively recognize and classify specific programs based on their syntax trees.

Code2Vec accomplished this by training models capable of recognizing well-known algorithms, such as sorting, searching, or counting – across multiple programming languages, including Java, C++ and Python. This approach involves generating a syntax tree from the source code, which is then analyzed by the trained model to identify the underlying algorithm. After the analysis, the model also provides a probability score for its prediction.

One notable example of applying Graph Neural Networks to program code analysis is Devign [10], in which the authors construct a code property graph (CPG) by merging Abstract Syntax Trees (AST), Control Flow Graphs (CFG), and Data

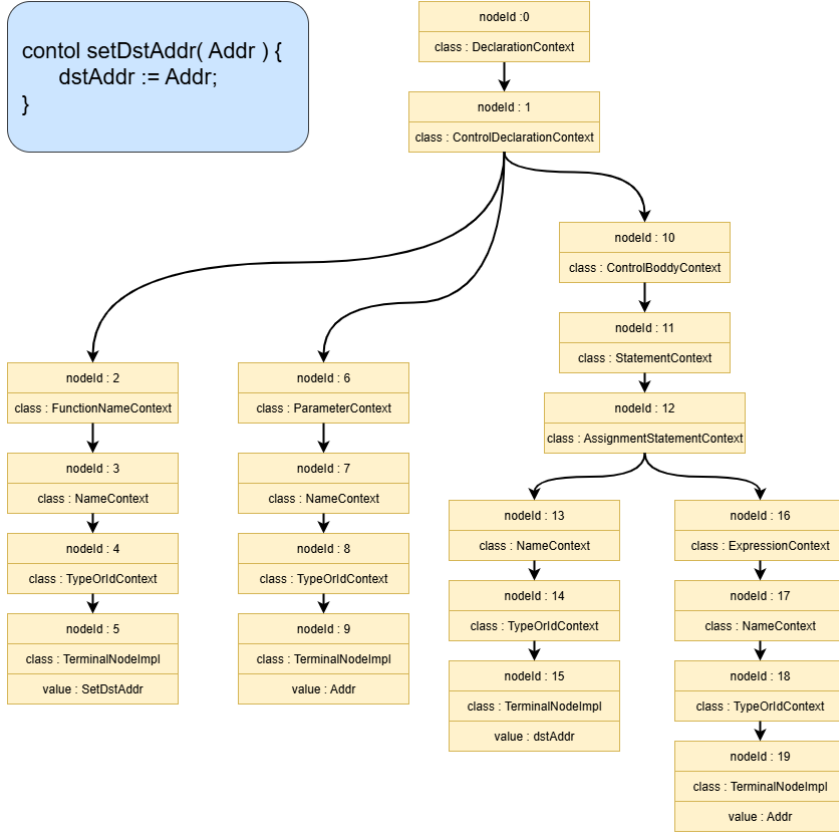


Figure 1. P4 function implementation and its AST.

Flow Graphs (DFG) into a single rich representation. The model – built upon a Gated Graph Neural Network – learns comprehensive program semantics from these graphs to identify security vulnerabilities in real world C code. In empirical evaluations on large open source projects, Devign significantly outperformed previous state-of-the-art vulnerability detection methods, improving accuracy by over 10% and F1 score by approximately 8-9%. This work exemplifies how merging multiple program structure views and leveraging GNNs enables robust, semantics aware analyses, and inspires similar approaches in tasks like variable renaming or parameter reordering in P4 ASTs.

A closely related work is the paper titled *P4 Specific Refactoring Steps* [8], which introduces a set of rule based refactoring operations specifically tailored for P4 programs. The approach is built on the P4Query framework, utilizing its syntactic and semantic graph based representations to perform transformations on abstract syntax trees (ASTs). These refactorings include table splitting, merging, and exe-

cution reordering, all governed by static analysis and predefined preconditions to ensure semantic correctness. While this method is based on manually designed transformations, our work extends this line of research by applying machine learning – specifically Graph Neural Networks – to automatically learn transformation patterns from data. This enables a shift from rigid, rule based systems toward adaptive, data driven refactoring strategies that generalize across diverse P4 programs.

3. Methodology

3.1. Graph construction from P4 programs

To obtain structured representations of P4 programs, we rely on P4Query, a comprehensive analysis tool that extracts detailed abstract syntax trees (ASTs) and other intermediate structures from P4 source code. The tool uses the official P4 language protocol to generate a standardized and richly annotated AST, which is interpreted as a large directed graph. The AST’s structure is defined by the syntactic rules of the language, and its edges encode diverse semantic relations; for instance, control flow and data flow dependencies.

Each node in the graph has several attributes that capture both syntactic and positional information. These include a unique `nodeId`, a `line` attribute indicating the location in the source code, `begin` and `end` fields that mark the span of the syntactic construct, a `type` field (e.g., `TerminalNodeImpl`), and, in the case of terminal nodes, a `value` representing the literal content. When preparing this graph for training a Graph Neural Network (GNN), we encode only the `type` and `value` attributes (if present) to create initial node feature vectors. This selective encoding strikes a balance between expressiveness and efficiency, allowing the model to focus on meaningful structural relationships while leveraging the inductive bias of the syntax driven graph topology.

3.2. Training of the GNN

To train our Graph Neural Network (GNN) models, we first generated several different P4 abstract syntax trees (ASTs) using P4Query. Each AST is treated as a directed graph, representing the dataset. As described earlier, each node’s attributes have to be auto encoded using its `type` and (if available) `value` attributes, producing a lightweight yet expressive node feature representation suitable for neural processing.

During the training procedure each epoch, the GNN is presented with a modified version of a graph and tasked with reconstructing the original structure. During the first epoch, the full graph is shown to the model, establishing a baseline understanding of the node representations and their connectivity. From the second epoch onward, we progressively apply structured degradation to the input graphs. Initially, we remove only terminal nodes (i.e., nodes of type `TerminalNodeImpl`),

which are typically leaves of the AST. In later stages, the removal process targets increasingly complex subgraphs, thereby challenging the model to infer more abstract syntactic patterns.

Importantly, the complexity of the removed subgraphs increases exponentially as training progresses. That is, in each subsequent training phase, the size and structural depth of the deleted subgraphs grow according to an exponential schedule. For example, while early stages may remove only isolated leaf nodes or shallow branches, later phases may eliminate entire nested control structures, such as `if else` blocks, loops, or parameter lists. This exponential degradation ensures a curriculum style learning process in which the model first learns to reconstruct simple local patterns, and only later faces the challenge of recovering deeply structured and semantically rich fragments of the AST.

The goal of this curriculum style training is to gradually force the model to internalize deeper compositional structures in the input graphs. By learning to predict missing parts of the AST across a variety of P4 programs, the GNN develops a generalizable representation of the language’s syntactic structure. Throughout training, the model is evaluated based on its reconstruction accuracy, which reflects how well it can recover node identities and their connections.

Using this training strategy, we developed three distinct GNN models, each targeting a specific code transformation or analysis task: a variable renamer, a parameter reorderer, and an empty `else` block detector. Although these models share the same underlying graph based training framework, each addresses a different aspect of code semantics.

These tasks were carefully selected to reflect realistic challenges in network code development. By automating variable renaming, argument ordering, and dead code detection, the models target areas where inconsistency and redundancy frequently arise in practice. Their utility lies not only in code transformation, but also in establishing and enforcing coding conventions across diverse teams and projects.

3.3. Variable renamer

The Variable Renamer model is trained to recognize and rename variables across abstract syntax trees (see an example in Listing 1). It is implemented as a two layer Graph Neural Network (GNN), which performs node classification over the AST graph. Each node is represented by a simple feature vector encoding whether it is a terminal node and whether it contains a value. The model learns to predict the original `value` attribute of `TerminalNodeImpl` nodes based on their surrounding context.

Listing 1. Renaming variables in P4 source code.

```
header ethernet_t {
    macAddr_t dstAddr;
    macAddr_t srcAddr;
    bit<16> etherType;
}
```

```

header ethernet_t {
    macAddr_t destinationAddr;
    macAddr_t sourceAddr;
    bit<16> ethernetType;
}

```

During training, graphs are labeled by the original variable names of terminal nodes, which serve as supervision targets. Once trained, the model takes as input a complete AST graph and a variable name, and it modifies the **value** attribute of all relevant nodes to a new name. This allows for consistent refactoring, such as renaming **egress_spec** to **egress_specific** across all its occurrences in the graph. After 20 epochs of training, the model successfully learned the graph structure and renaming patterns by leveraging structural differences between input graphs.

The renaming decision is not based solely on the textual value of a node, but also on its structural context. This enables the model to generalize and identify semantically equivalent nodes in different programs, even when local syntax varies. The GNN's message passing mechanism allows it to aggregate information from neighboring nodes, making the renaming behavior robust and context aware.

3.4. Parameter reorderer

The Parameter Reorderer model is designed to learn the correct ordering of function parameters in P4 programs. Listing 2 shows an example of the transformation. Unlike the Variable Renamer, this model is not a traditional GNN architecture. Instead, it uses a learned embedding layer in combination with a small feed forward neural network to predict the target position of each parameter.

The training is performed using pairs of AST graphs: one representing the original parameter order, and one representing the desired (reordered) form. Each parameter is represented by its subtree in the AST, and the model learns to assign a position score to each based on its embedded identity. The loss function minimizes the mean squared error between the predicted positions and the ground truth permutation derived from the reordered graph. The model required 100 epochs of training to learn the parameter reordering logic from structural graph differences.

Once trained, the model receives a new AST and infers a new permutation of the parameters. It then rewrites the structure of the graph accordingly, updating the 'start' and 'end' positions of the affected nodes and regenerating comma separators where needed. This results in a syntactically valid AST that reflects the learned parameter order.

Listing 2. Reordering P4 function's parameter order.

```

control MyDeparser(packet_out packet, in headers hdr) {
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.ipv4);
    }
}

```

```

    }

    control MyDeparser(in headers hdr, packet_out packet) {
        apply {
            packet.emit(hdr.ethernet);
            packet.emit(hdr.ipv4);
        }
    }
}

```

The model’s ability to operate directly on AST node embeddings without requiring handcrafted rules – enables flexible reordering strategies and supports the standardization of function signatures across large codebases.

3.5. Empty Else Block Detector

The Empty Else Block Detector is a binary classification model built to detect empty `else` branches (for example, those shown in Listing 3) in P4 program ASTs. It is implemented as a two layer GNN, trained to classify specific nodes labeled as `else` into two categories: empty or nonempty.

The model is trained on a dataset of P4 ASTs, where each graph is automatically labeled using a rule based search for syntactically empty `else` branches. During preprocessing, only nodes with the literal value "`else`" are included in the training set, and are labeled according to whether they lead to an empty block in the AST. Each node is encoded using a pair of categorical features: the class of the node (e.g., `TerminalNodeImpl`, `StatementContext`) and the textual value (if present). The model was trained for 100 epochs to capture the subtle structural patterns associated with empty else branches.

The GNN learns to classify these `else` nodes based on their surrounding context in the AST. Its performance is measured via accuracy on the classification task. Once trained, the model can analyze unseen ASTs and assign a confidence score to each `else` branch, indicating whether it is likely to be empty.

This model serves as a useful tool for detecting redundant or misleading conditional structures in P4 programs. By identifying empty branches that serve no semantic purpose, the model supports further code cleanup and optimization.

Listing 3. P4 if block with empty else

```

if (hdr.ipv415.isValid()) {
    if (hdr.ipv416.isValid()) {
        ipv415.lpm.apply();
        ipv416.lpm.apply();
    } else {}
} else {}

```

4. Results

All models presented in this work were implemented using the `PyTorch` [6] and `PyTorch Geometric` libraries, which provide efficient GPU accelerated operations and high-level abstractions for graph based deep learning. These libraries enabled rapid experimentation with neural architectures and streamlined the construction and training of Graph Neural Networks. We used standard components such as `GCNConv` layers, embedding modules, and training utilities provided by the framework.

In our experiments, the abstract syntax trees (ASTs) generated by the P4Query [5] tool were serialized as large JSON files. Each file contains a directed graph with thousands of nodes and edges, representing detailed syntactic and semantic information of a given P4 program. Due to the high complexity and richness of these graphs, even a small number of training examples proved sufficient for effective learning.

We observed that all three models trained on these AST graphs exhibited consistently decreasing loss values throughout training, indicating convergence and successful pattern extraction. The loss function approached values close to zero across multiple epochs, demonstrating that the models were able to learn the structural relationships within the graphs efficiently.

4.1. Variable Renamer results

The Variable Renamer model was trained on a set of 32 AST graphs, each representing a different P4 program. These graphs served as supervised input, where variable renaming labels were provided for terminal nodes. Training was performed over 20 epochs, which proved sufficient for the model to converge – the loss function consistently decreased and reached zero by the end of training. This indicates that the GNN successfully learned to encode the structure of a given variable’s representation, identifying the relevant nodes and edges responsible for that variable throughout the graph.

Once trained, the model was evaluated on 17 previously unseen AST graphs. In all cases, the model was instructed to rename a variable per graph. The model successfully performed all 17 renamings, modifying the corresponding `value` attributes in the appropriate `TerminalNodeImpl` nodes. Importantly, the renamings were consistent across all occurrences of the target variables within each graph.

These results highlight the model’s ability to generalize renaming patterns across different P4 codebases, even when local syntactic variations are present. This confirms the effectiveness of GNN based learning on AST structures.

4.2. Parameter Reorderer results

To evaluate the effectiveness of the Parameter Reorderer model, we trained it on 5 pairs (a total of 10) of P4 AST graphs. Each pair consisted of an original function declaration with an incorrect or arbitrary parameter order, and a corresponding

target graph in which the parameters were reordered according to a preferred or canonical sequence. Unlike the Variable Renamer, this model had to capture and interpret more complex structural patterns, as parameters are represented by entire subgraphs within the AST. As a result, learning the correct transformations required more epochs (100 in total), and convergence was slower.

In cases where the function had only two parameters, the model achieved correct reordering in 3 of the 5 example graphs, which were previously unseen ASTs. These are effectively binary swaps, which the model was often able to predict reliably. However, as the number of parameters increased, the model's accuracy declined. This was particularly evident in more complex declarations, where parameters had compound types or nested substructures. In such cases, the model struggled to infer the correct order, indicating a sensitivity to syntactic complexity.

These results suggest that while the model captures some general patterns of parameter ordering, it may lack sufficient contextual awareness to handle more elaborate structures. The approach, based on embedding and regression, shows promise for approximating permutation tasks but could benefit from stronger relational reasoning.

From these observations, we conclude that parameter reordering is more effectively approached as a regression problem over continuous embeddings than as a classification problem over discrete graph structures. Future improvements may involve incorporating structural attention or hybrid GNN-MLP architectures to better capture the interplay between parameter semantics and graph topology.

4.3. Empty Else Block Detector results

The Empty Else Block Detector model was trained on a dataset of 40 AST graphs, each annotated with both empty and nonempty `else` branches. During training, the model performed binary classification on nodes with the literal value `"else"`, predicting whether they led to an empty code block. Due to the complexity of the task – which required the model to learn to recognize non trivial subgraph patterns around each `else` node – the training process was conducted over 100 epochs. This extended training was necessary for the model to reliably capture the subtle structural cues associated with empty branches. The final training accuracy reached 78%, demonstrating the model's ability to distinguish between semantically relevant and irrelevant `else` constructs.

To evaluate the model's generalization capability, we tested it on 13 previously unseen P4 programs, whose corresponding ASTs contained a total of 28 `else` blocks, including both empty and non-empty cases. The model correctly classified 23 out of the 28 instances, resulting in an 82% accuracy on this test set. While this reflects a slight increase compared to its training performance, the result is promising given the structural variability of `else` constructs in real world P4 code.

These results suggest that the model successfully captures the contextual signals that distinguish empty `else` branches from meaningful ones. Its ability to detect redundant conditional structures can support automated refactoring pipelines and contribute to cleaner, more maintainable P4 codebases.

Beyond demonstrating technical feasibility, our results indicate that GNN based learning provides a scalable path toward automated code improvement. Even with limited training data, the models extracted robust transformation patterns, suggesting practical applicability in real world P4 codebases, where manual refactoring is costly and time consuming.

5. Conclusion and future work

In this research, we have successfully developed three distinct Graph Neural Network (GNN) models: the Variable Renamer, the Parameter Reorderer, and the Empty Else Block Detector. Each of these models addresses specific aspects of P4 program refactoring, showcasing the versatility and potential of GNNs in this domain.

The Variable Renamer model receives the complete abstract syntax tree (AST) of a program as input and is tasked with renaming a specific variable throughout all its occurrences. It takes as parameters the graph representing the program and the names of the variable to be renamed along with the new name. By modifying the attributes of the relevant nodes in the AST, this model enhances code readability and consistency, which are crucial for maintaining large scale software systems.

The Parameter Reorderer model operates at the level of function declarations, where it modifies the structure of the graph to rearrange the order of parameters. This capability is particularly beneficial for adhering to coding standards or managing default parameters, ultimately improving the clarity and readability of function definitions. By ensuring that parameters are organized in a logical manner, this model aids developers in writing more maintainable and understandable code.

The Empty Else Block Detector model focuses on prediction tasks within the graph. It identifies patterns that indicate the presence of empty else branches, which can be particularly useful for code optimization. By alerting developers to these non essential branches, the model helps avoid unnecessary complexity in the code, thereby enhancing overall code quality and maintainability.

Through the development and evaluation of these models, we have gained valuable insights into how GNNs can learn to capture and manipulate different aspects of graph data. This experience lays the groundwork for future research, where we aim to generalize these findings to create more complex models that integrate the strengths of each individual approach. By combining the capabilities of modifying node attributes, altering graph structures, and making predictions, we envision the development of sophisticated GNN architectures that can tackle a wider range of refactoring tasks.

The strength of graph neural network models, compared to traditional algorithms, lies in their ability to generalize the structure of abstract syntax trees through training on large datasets. Embeddings and heuristics play a major role in this process. As a result, these models can perform their tasks effectively and reliably, even in extreme cases. In contrast, traditional algorithms must explicitly account for all such edge cases, which requires more development time. This can

increase the algorithm's size, complexity, or reduce its clarity. Furthermore, when a new edge case is discovered, the original algorithm must be revised or rewritten, further increasing development time and costs. Ultimately, although GNN models typically require more storage space and computational power during execution, they can successfully handle a wider range of situations than traditional approaches.

Looking ahead, our goal is to explore the potential of hybrid models that leverage the strengths of the existing GNNs while introducing new mechanisms for learning and adaptation. Such models could incorporate advanced techniques such as attention mechanisms or reinforcement learning to further enhance their performance and applicability. By pushing the boundaries of what GNNs can achieve in the context of program analysis and refactoring, we hope to contribute to the creation of more intelligent and automated tools that assist developers in writing high quality, maintainable code. By integrating these models into tooling pipelines, development teams can achieve faster iteration cycles, reduce manual overhead, and promote uniform, high quality code across network applications.

References

- [1] U. ALON, M. ZILBERSTEIN, O. LEVY, E. YAHAV: *code2vec: learning distributed representations of code*, Proc. ACM Program. Lang. 3.POPL (Jan. 2019), DOI: [10.1145/3290353](https://doi.org/10.1145/3290353).
- [2] P. BOSSHART, D. DALY, G. GIBB, M. IZZARD, N. MCKEOWN, J. REXFORD, C. SCHLESINGER, D. TALAYCO, A. VAHDAT, G. VARGHESE, D. WALKER: *P4: programming protocol-independent packet processors*, SIGCOMM Comput. Commun. Rev. 44.3 (July 2014), pp. 87–95, ISSN: 0146-4833, DOI: [10.1145/2656877.2656890](https://doi.org/10.1145/2656877.2656890).
- [3] J. JONES: *Abstract Syntax Tree Implementation Idioms*, Technical Report, PDF, University of Alabama, 2003, URL: <https://hillside.net/plop/plop2003/Papers/Jones-ImplementinGASTs.pdf>.
- [4] A. KHERADMAND, G. ROŞU: *P4K: A Formal Semantics of P4 and Applications*, arXiv preprint (2018), arXiv:1804.01468.
- [5] D. LUKÁCS, G. TÓTH, M. TEJFEL: *P4Query: Static analyser framework for P4*, Annales Mathematicae et Informaticae 57 (2023), pp. 49–64, DOI: [10.33039/ami.2023.03.002](https://doi.org/10.33039/ami.2023.03.002).
- [6] A. PASZKE, S. GROSS, F. MASSA, A. LERER, J. BRADBURY, G. CHANAN, T. KILLEEN, Z. LIN, N. GIMELSHEIN, L. ANTIGA, A. DESMAISON, A. KÖPF, E. YANG, Z. DEVITO, M. RAISON, A. TEJANI, S. CHILAMKURTHY, B. STEINER, L. FANG, J. BAI, S. CHINTALA: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, arXiv:1912.01703, NeurIPS 2019, 2019, URL: <http://arxiv.org/abs/1912.01703>.
- [7] M. A. RODRIGUEZ: *The Gremlin graph traversal machine and language (invited talk)*, in: Proceedings of the 15th Symposium on Database Programming Languages, DBPL 2015, Pittsburgh, PA, USA: Association for Computing Machinery, 2015, pp. 1–10, ISBN: 9781450339025, DOI: [10.1145/2815072.2815073](https://doi.org/10.1145/2815072.2815073).
- [8] M. TEJFEL, D. LUKÁCS, P. HEGYI: *P4 Specific Refactoring Steps*, Acta Cybernetica 27.1 (Mar. 2025), pp. 53–65, DOI: [10.14232/actacyb.308085](https://doi.org/10.14232/actacyb.308085), URL: <https://cyber.bibl.u-szege.d.hu/index.php/actcybern/article/view/4465>.
- [9] J. ZHOU, G. CUI, S. HU, Z. ZHANG, C. YANG, Z. LIU, L. WANG, C. LI, M. SUN: *Graph neural networks: A review of methods and applications*, AI Open 1 (2020), pp. 57–81, ISSN: 2666-6510, DOI: [10.1016/j.aiopen.2021.01.001](https://doi.org/10.1016/j.aiopen.2021.01.001), URL: <https://www.sciencedirect.com/science/article/pii/S2666651021000012>.

- [10] Y. ZHOU, S. LIU, J. SIOW, X. DU, Y. LIU: *Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks*, in: Advances in Neural Information Processing Systems (NeurIPS), 2019.

Artificial Intelligence based robotic applications for higher education*

Isabela Drămnesc^a, Erika Ábrahám^b, László Antal^b,
Péter Csóka^f, Nikolaos Fachantidis^c, Tudor Jebelean^a,
Gábor Kúspér^{d,e}, Konstantinos Papadopoulos^c

^aWest University of Timișoara, Romania
{isabela.dramnesc,tudor.jebelean}@e-uvt.ro

^bRWTH Aachen University, Germany
{abraham,antal}@cs.rwth-aachen.de

^cUniversity of Macedonia, Thessaloniki, Greece
{nfachantidis,kostasap}@uom.edu.gr

^dEszterházy Károly Catholic University, Hungary
kusper.gabor@uni-eszterhazy.hu

^eUniversity of Debrecen, Hungary
kusper.gabor@inf.unideb.hu

^fInnovITech Ltd., Hungary
peter.csoka@innovitech.hu

Abstract. This paper presents some key outcomes of the European project AiRobo, a collaborative initiative involving five universities from Romania, Germany, Greece, Hungary, and France. This is an educational project targeting higher education students, academic staff, and industry professionals. This paper presents four real-world AI-based robotic applications designed to serve as engaging teaching support materials, particularly for theoretical courses that students often find challenging to grasp. These practical applications are designed to bridge the gap between theory and practice, simplifying complex concepts and making them more accessible. They also aim to enhance the learning and teaching process, making it more engaging, motivating, and appealing for both students and academic staff. We present four real-world AI-based robotic applications: two applications integrating

*This work is co-funded by the EU through the Erasmus+ project *AiRobo: Artificial Intelligence-based Robotics*, 2023-1-RO01-KA220-HED-000152418.

ChatGPT in Pepper and NAO robots are designed to serve as learning assistants for students, and two applications (one with the simulation of an agriculture robot in Unity and an AI-based underwater robot controller) are designed to serve as engaging teaching materials on AI-related subjects, but also for theoretical courses that students often find challenging to grasp. For each application, we outline its scope, model, and implementation code. In addition, the implementation code will be openly accessible in the final version of the paper, enabling academic staff and researchers to easily use and adapt these case studies for their own educational or research tasks.

Keywords: Artificial Intelligence, robotics, higher education

AMS Subject Classification: 68T40 Artificial intelligence for robotics

1. Introduction

Robots are indispensable across industries, from manufacturing and agriculture to research and healthcare, with AI further expanding their capabilities. Improving the quality of education in related areas is important for the development of these technologies and of the society in general. In education, applications as demonstrators and illustrators are of utmost importance, but are rarely available.

In this paper, we present four applications we developed that may be useful for researchers and teachers in subjects related to artificial intelligence, robotics, and formal verification (for example: computational and mathematical logic, automated theorem proving, algorithm synthesis and mathematical theory exploration, modeling and certifying algorithms, verification of hybrid systems, satisfiability checking, machine learning, etc.), as well as serving as learning assistants for students. The first two applications are AI-enabled robots to personalize learning and serve as teaching assistants, enhancing academic outcomes. Two further applications are AI-controlled robots that can serve as educative case studies, for example in machine learning, or for teaching formal methods to ensure safety and reliability.

Robots as student assistants. AI and robotics may enhance students' cognitive abilities through brain-computer interfaces (BCIs) [16–18]. Social robots [8] have the potential to actively engage students and enrich their learning experience. For instance, *chatbots* can interact through text or voice using AI [23, 30]. A comprehensive literature review for the use of AI in education is [32], which also reviews the use of robots [21].

As a first application, in Section 2 we describe the *Pepper robot* [28], which offers significant potential for improving student engagement, particularly in language acquisition [2, 9]. ChatGPT has been used in education [11], as well as in robotics applications [31]. The study [27] examines how integrating large language models (LLMs), such as Google PaLM2 and ChatGPT, into the Pepper robot, in conjunction with Reinforcement Learning with Human Feedback (RLHF), can enhance the robot's natural language processing (NLP) capabilities.

Our second application presented in Section 3 is a NAO storyteller. The NAO robot [29] can serve as an interactive teaching assistant [15], interactive storyteller [10], catalyst for teamwork [33], and others [7].

Robots as use cases. There is not much material available to support teaching in subjects related to AI-based robotics. Some loosely related research on the importance of logic in higher education [24] and on teaching applied formal methods [25] both address the verification of cyber-physical systems.

To provide further support, in Section 4 we present a plant-watering robot application. There are very few tutorials or educational material available in the field of agricultural robotics. For instance, [13] compares Gazebo and Unity for digital twin simulation in the field of agriculture, and [19] describes a robotic application using IoT (Internet of Things) for the intelligent watering of plants. Addressing agricultural harvesting, [12] describes a pneumatic gripper for robotic use inspired by gecko's foot and human finger.

Our last application, presented in Section 5, is an autonomous underwater vehicle. It can be used to illustrate, e.g., data gathering and processing, machine learning, simulation, testing, and formal verification.

2. Student assistance: Pepper with ChatGPT

Use case. The integration of AI assistants into classrooms has opened new possibilities for student engagement, comprehension, and skill development. However, many learners struggle with using AI-powered tools effectively due to a lack of context or familiarity. We programmed the Pepper robot as an interactive learning assistant, designed to support students in their academic journey, while fostering an understanding of AI-driven tools like ChatGPT.

Methodology. When Pepper detects human presence, it introduces itself and offers assistance to the user. It invites them to ask questions and provides relevant support. If the user provides sufficient context for ChatGPT to confidently generate a clear and accurate response, Pepper reads out the answer and congratulates the user. However, if the input lacks context, Pepper prompts the user to rephrase their question with more details to ensure a more precise and meaningful response. Thus Pepper assists by demonstrating how AI can be a valuable educational resource, explaining its strengths and limitations, while guiding students to frame their queries for better responses.

Implementation. Executing applications on Pepper requires certain workarounds due to the limitations of its underlying software. Pepper runs on NAOqi 2.5, an outdated framework that supports only Python 2.7, which introduces constraints, particularly in multi-lingual environments due to long-standing issues with Python

2.7. This creates challenges when integrating modern AI models such as ChatGPT. To enable ChatGPT functionality on Pepper, an external proxy for ChatGPT requests is required to handle requests and process responses efficiently for ChatGPT to function on Pepper’s hardware. Pepper’s Automatic Speech Recognition (ASR) system has inherent limitations, particularly with free-text input and multilingual speech, which led us to employ an external transcription service. In our implementation we used Google’s Cloud Speech-to-Text API, selected for its higher recognition accuracy, stable latency (about 1-1.5 seconds in classroom use), and straightforward integration with the Python proxy service. An audio stream of student questions were transmitted to the API for transcription; to address privacy and data protection concerns, no personal identifiers were included in the recordings, access was limited to authorized users, and API usage was logged for auditing. This transcription is then forwarded to the ChatGPT API, where the response is generated externally and relayed back to Pepper for delivery to the student.

These adjustments ensure that Pepper can assist students despite its software constraints.

The source code for the robot is openly accessible at: <https://github.com/KostasPapadopoulosUOM/AiRobo/tree/main/PepperAIAssistant>. Our implementation requires a ChatGPT API Key. The end user will need to create an OpenAI account and then generate an API key. The API key is required to be placed at the following path: `/data/home/nao/chatgpt.key`. An external ASR engine is also required due to the before mentioned limitation. In this case Google’s Transcription API is used, which requires an additional Google API key with the *Cloud Speech-to-Text API* enabled. The end user will need to place the API Key in `/data/home/nao/googleapi.key`.

Teaching. A significant challenge in AI-assisted education is ensuring that students develop critical thinking skills and understand the importance of context in learning. Pepper facilitates discussions that encourage students to ask meaningful questions and refine their approach when using AI tools. By doing so, it ensures that students do not rely on AI blindly but instead use it as a complementary tool to their own reasoning and problem-solving abilities.

3. Student assistance: Nao storyteller

Use case. We further leverage the NAO robot’s capabilities to create a dynamic learning experience that fosters curiosity and engagement among students. By integrating AI-driven storytelling and vision recognition, we aim to enhance students’ understanding of both robotics and AI in an interactive manner.

Methodology. The NAO robot operates using its autonomous life mode, allowing it to exhibit human-like behaviors such as looking around, adjusting its posture, and responding naturally to stimuli. These features help to create a more immersive and interactive experience for students. The key components of our methodol-

ogy include computer vision, AI-generated storytelling, and embodied interaction through gestures and speech. Additionally, we employ text-to-speech technology to ensure the robot delivers narratives in a clear and expressive manner, and automatic speech recognition to allow students to interact with the robot through spoken commands, further bolstering engagement.

Implementation. The NAO robot was programmed with the following functionalities. (i) Face Detection: the robot identifies and acknowledges the presence of students, creating a more personalized interaction. (ii) Vision Recognition: using its onboard camera, NAO recognizes images it has been trained on. In this study, we employed cubes featuring images of ancient Greek gods. (iii) Storytelling via AI: upon recognizing an image, the robot queries an AI system to generate a humorous story about the identified Greek god. It then reads the story aloud while performing subtle movements to maintain engagement. The source code for this application is publicly available at: https://github.com/KostasPapadopoulosUOM/AiRobo/tree/main/NAO_Cube_Game.

This implementation requires a ChatGPT API key. To obtain one, the end user must create an OpenAI account and generate an API key. The key should be stored at the following path: `/data/home/nao/chatgpt.key`. This configuration allows the NAO robot to interact with ChatGPT for generating educational content and storytelling responses.

Teaching. The primary objective of this project was to introduce students to AI and robotics in an engaging and interactive manner. By incorporating storytelling and physical interaction, the students were encouraged to explore AI capabilities while learning about ancient mythology. Observations suggest that the robot's interactive features improved student participation and interest.

4. Robots as use cases: Agriculture robot

Use case. In this section we present our plant-watering robot, whose control is specified by the state machine depicted on Figure 1. The robot starts at its charging station, where it recharges and refills its water tank before navigating to water flowers at predefined locations. It encounters obstacles, classified as living (e.g., animals) and non-living (e.g., rocks); and attempts to navigate around them. If avoidance is impossible, it requests external assistance. When battery or water levels are low, the robot returns to the charging station; if it depletes its energy before reaching the station, it also requests help. To simplify the state machine, we have a special state, called `ALL_STATES`, which describes the common behavior of each state. We used the hardware platform SCOUT MINI [1], equipped with GPS, a compass, two LiDAR sensors (front and rear), and a velocity sensor.

Methodology. The robot's navigation relies on reinforcement learning (RL), where the agent optimizes movement through action rewards, using the Proxi-

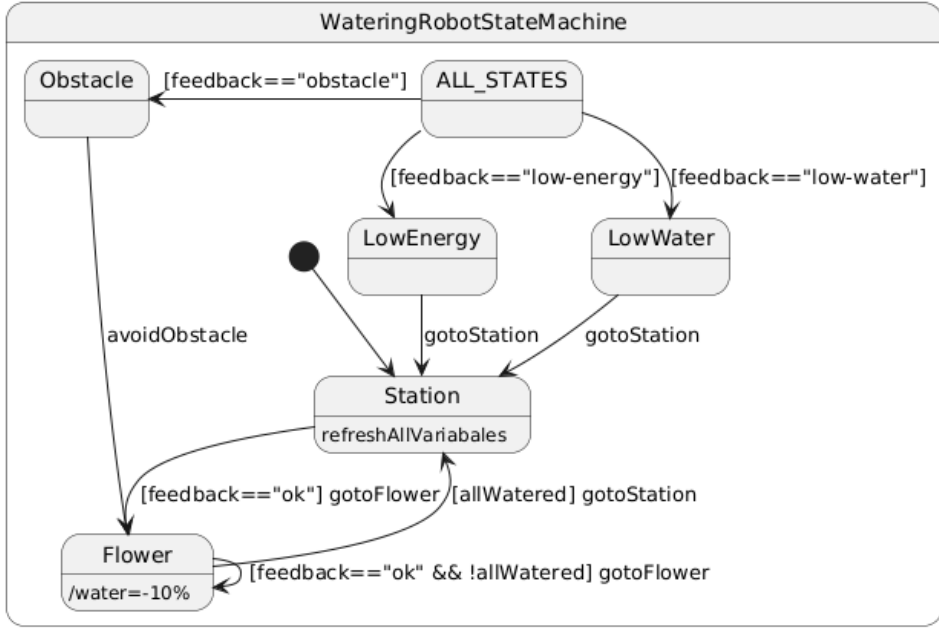


Figure 1. Finite state machine of the plant-watering robot.

mal Policy Optimization (PPO) algorithm. The hyperparameters used for the algorithm and training can be seen in Table 1. The observation space is an 11-dimensional continuous vector representing the robot's state. It includes the angular difference between the robot's orientation and the target vector, normalized to $[-1, 1]$. The linear velocity of the robot, also normalized to $[-1, 1]$. The distance between the robot and the target, normalized by the initial distance at the beginning of each episode to a range of $[0, 1]$. There are two LiDAR sensors on the robot. The forward facing sensor covers a 180° , from -90° to 90° relative to the robot's heading. It is divided into 4 equal sections, and the minimum reading is taken in each section. The process is mirrored for the rear sensor, resulting in a total of eight distance readings, each normalized to $[0, 1]$. The action space is a 2-dimensional continuous vector, defining the rover's linear and angular velocity, each component normalized to $[-1, 1]$.

The reward function is designed to guide the agent towards its target while avoiding obstacles. More formally, the reward R_t at each timestep t is defined as:

$$R_t = \begin{cases} +200 & \text{if } d_t < \epsilon_{\text{target}} \quad (\text{goal reached}) \\ -200 & \text{if collision occurs} \\ -150 & \text{if } t \geq T_{\text{max}} \quad (\text{timeout}) \\ 10 \cdot (d_{t-1} - d_t) & \text{otherwise} \end{cases}$$

Table 1. List of hyperparameters.

Hyperparameter	Value
Algorithm Parameters	
Number of epochs	8
Batch size	1024
Discount (γ)	0.99
GAE parameter (λ)	0.95
PPO clip range (ϵ)	0.2
Critic Loss Coefficient	0.5
Network Architecture	
Number of hidden layers	2
Hidden layer size	128
Hidden Layer Activation	ReLU
Actor Output Activation	Tanh
Optimizer algorithm	Adam
Training parameters	
Initial learning rate	3e-4
Learning rate decay	linear decay
Learning rate decay frequency (in episodes)	250
Minimum learning rate	1e-8
Standard deviation of selected actions	1

Here, d_t is the distance to the target at timestep t , ϵ_{target} is the success threshold distance set to 0.1 meters, and T_{max} is the maximum episode length, set to 1500 steps. A penalty is applied immediately upon any collision. The term $(d_{t-1} - d_t)$ provides a reward for moving closer to the target and a penalty for moving away in non-terminal states.

To enhance training, we apply curriculum learning [22], gradually increasing task complexity. RL agents often struggle with generalization. To address this, elements of the training environment are randomized.

The first stage involves point-to-point navigation. Here, the environment is a 10×10 meter plane. For each episode, the position of the target and the starting pose of the robot are randomly selected within this area. During this phase, the RL agent only uses the relative heading, velocity and distance to the target as observations. The training continues until 90 out of the last 100 episodes are successful.

The second stage introduces obstacles in the form of walls, to teach avoidance behavior. These walls are parallel to the direct path to the target and are positioned to form a 1.5-meter-wide corridor. The length of each wall is randomized for each episode, ranging from 0.1 to 15 meters. Here, the agent uses the full observation

vector including LiDAR data.

The final stage is a randomly generated but static garden resembling real-world conditions. The walls are replaced by randomly placed flowers. There is a moving dog to test the agent’s robustness to changing conditions. It is used to evaluate the RL agent’s behavior in a previously unseen and more realistic setting.

Catastrophic forgetting [14] is another challenge, where learning a new task degrades previously learned behaviors. This is mitigated by periodically revisiting earlier tasks, ensuring stable performance across different environments. Specifically, during the second stage, the environment alternates between the first and second stage every 100 episodes.

Implementation. The system consists of a Unity-based simulation and a Python module for RL training and testing. Running the simulation requires installing Unity. The final implementation is publicly available at <https://github.com/csokapeter/Agricultural-Robot>.

Teaching. This project serves as an educational tool for reinforcement learning and robotics. Students gain hands-on experience with RL-based navigation, digital sensor integration in Unity, and training methodologies. It allows them to experiment with reward functions, agent parameters, and real-world deployment challenges. Additionally, it highlights hardware limitations and safety considerations when transferring trained agents from simulation to physical robots.

5. Robots as use cases: Underwater robot controller

Use case. Autonomous underwater vehicles (AUVs) provide a promising technique to perform underwater exploration missions autonomously [4, 5]. To successfully operate in an uncertain, a priori unknown environment, a *rule-based* algorithm proposed in [20] relies on specific sensor measurements – such as depth, altitude and forward distance – to estimate the steepness of the seafloor (α), allowing the robot to adjust its pitch (β) accordingly (see Figure 2). Using these sensor values, *bottom tracking* maintains the distance to the seafloor as constant as possible, to increase the reliability of sensor data gathering, and *obstacle avoidance* recognizes rocks, walls etc. to avoid collisions.

While this method performs reasonably well in smooth environments, it is highly sensitive to noise, and it requires highly complex code, making it difficult to debug and maintain. To address these limitations, in [3] the authors propose an alternative *AI-based* controller to overcome these challenges.

Methodology. The AI-based controller [3] utilizes a neural network, which uses the observed sensor values within some time window to issue control commands within a long short-term memory (LSTM) architecture (see Figure 2). These control commands adjust the pitch of the AUV, facilitating bottom tracking with more efficient obstacle avoidance (less braking) and better robustness.

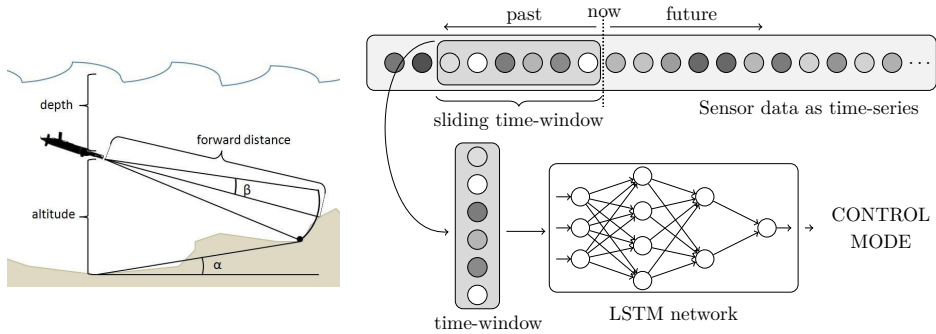


Figure 2. Left: Illustration from [20] for the *rule-based* obstacle avoidance method. Right: The LSTM controller system architecture.

A key challenge of [3], is the lack of *labeled* data for the training. To address this, the authors used sensor data of logs from previously executed real world missions, and employed signal processing techniques for labeling each timestamp. Further improvements were achieved through online re-training using a simulator, and the employment of a simplex architecture [6, 26] to revert to the rule-based method in case the AI-based control would not prevent a collision. The controller was deployed on an AUV of OceanScan MST and two real world surveys were conducted. These demonstrated that the AI-based controller produced better efficiency (i.e., shorter mission time, less battery usage), while maintaining safety (no collisions) throughout the mission.

Implementation. The code implementation, the trained neural network in ONNX format, and further instructions on how to setup and use the simulator with the neural network controller is openly accessible at: <https://github.com/antalla szlo011/improved-AUV-obstacle-avoidance>.

Teaching. This application can be used to illustrate the *development of AI-based controllers*, including the preparation of training data, the training and re-training of neural networks, knowledge distillation, and the embedding of AI-based controllers in a simplex architecture for fall-safe functioning. Besides simulation and testing, *formal methods* can be applied to assess the reliability and safety of the AUV behavior.

6. Conclusions

In this paper we presented four real-world, AI-based robotic applications specifically designed to enhance the learning experience for students. These applications not only serve as interactive assistants, making the learning process more engaging,

but also provide practical solutions for complex theoretical courses that students often struggle with, helping them grasp challenging concepts more effectively. A key contribution of this work is the free availability of the application implementations, empowering academic staff and researchers to seamlessly integrate or adapt these resources into their own teaching or research activities.

As future work, the authors consider: Integrating various generative AI engines, such as DeepSeek, which offers the advantage of self-hosting at no cost; Incorporating alternative transcription engines with a focus on self-hosted solutions, potentially leveraging OpenAI's Whisper model, as well as evaluating different deployment strategies and optimizing model performance for real-time transcription in educational contexts; Implementing additional sensors in the agricultural robot simulation to better reflect real-world conditions and allow students to experiment with different configurations of the robot.

References

- [1] AGILE X: *SCOUT MINI*, <https://global.agilex.ai/products/scout-mini>.
- [2] M. ALEMI, A. MEGHDARI, M. GHAZISAEDY: *The Impact of Social Robotics on L2 Learners' Anxiety and Attitude in English Vocabulary Acquisition*, eng, International Journal of Social Robotics 7 (2015), pp. 523–535, DOI: [10.1007/s12369-015-0286-y](https://doi.org/10.1007/s12369-015-0286-y).
- [3] L. ANTAL, M. AUBARD, E. ÁBRAHÁM, A. MADUREIRA, L. MADUREIRA, M. COSTA, J. PINTO, R. CAMPOS: *A Collision Avoidance Method for Autonomous Underwater Vehicles Based on Long Short-Term Memories*, in: Proc. of the 13th Innovations in Bio-Inspired Computing and Applications (IBICA'22), vol. 649, LNNS, Springer, 2023, pp. 448–457, ISBN: 978-3-031-27499-2, DOI: [10.1007/978-3-031-27499-2_42](https://doi.org/10.1007/978-3-031-27499-2_42).
- [4] M. AUBARD, L. ANTAL, A. MADUREIRA, E. ÁBRAHÁM: *Knowledge Distillation in YOLOX-ViT for Side-Scan Sonar Object Detection*, CoRR abs/2403.09313 (2024), DOI: [10.48550/ARXIV.2403.09313](https://doi.org/10.48550/ARXIV.2403.09313), arXiv: [2403.09313](https://arxiv.org/abs/2403.09313).
- [5] M. AUBARD, L. ANTAL, A. MADUREIRA, L. F. TEIXEIRA, E. ÁBRAHÁM: *ROSAR: An Adversarial Re-Training Framework for Robust Side-Scan Sonar Object Detection* (2024), DOI: [10.48550/ARXIV.2410.10554](https://doi.org/10.48550/ARXIV.2410.10554), arXiv: [2410.10554](https://arxiv.org/abs/2410.10554).
- [6] S. BAK, D. K. CHIVUKULA, O. ADEKUNLE, M. SUN, M. CACCAMO, L. SHA: *The system-level simplex architecture for improved real-time embedded system safety*, in: Proc. of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'09), IEEE, Institute of Electrical and Electronics Engineers, 2009, pp. 99–107, DOI: [10.1109/rta.2009.515501](https://doi.org/10.1109/rta.2009.515501).
- [7] H. BANAEIAN, I. GILANLIOĞLU: *Influence of the NAO robot as a teaching assistant on university students' vocabulary learning and attitudes*, Australasian Journal of Educational Technology (2021), pp. 71–87, DOI: [10.14742/ajet.6130](https://doi.org/10.14742/ajet.6130).
- [8] T. BELPAEME, J. KENNEDY, A. RAMACHANDRAN, B. SCASSELLATI, F. TANAKA: *Social robots for education: A review*, Journal of Science Robotics Vol. 3.Issue 21 (2018), DOI: [10.1126/scirobotics.aat5954](https://doi.org/10.1126/scirobotics.aat5954).
- [9] T. BELPAEME, F. TANAKA: *Social robots as educators*, in: OECD Digital Education Outlook 2021 Pushing the Frontiers with Artificial Intelligence, Blockchain and Robots: Pushing the Frontiers with Artificial Intelligence, Blockchain and Robots, OECD Publishing Paris, 2021, p. 143.

- [10] H. CAO, R. SIMUT, N. KREPEL, B. VANDERBORCHT, J. VANDERFAEILLIE: *Could NAO robot function as model demonstrating joint attention skills for children with autism spectrum disorder? An exploratory study*, International Journal of Humanoid Robotics 19 (04 2022), DOI: [10.1142/s0219843622400060](https://doi.org/10.1142/s0219843622400060).
- [11] O. E. CHINONSO, A. M.-E. THERESA, T. C. ADUKE: *ChatGPT for Teaching, Learning and Research: Prospects and Challenges*, Global Academic Journal of Humanities and Social Sciences (2023), URL: <https://api.semanticscholar.org/CorpusID:257476234>.
- [12] A. CLARK, L. GOODSSELL-CARPENTER, P. BUCKOW, D. HEWETT, F. WHITE, A. IMAM, N. NAZ, B. ROBINSON, S. K. MANNA, A. AHMED: *Bio-Inspired Soft Pneumatic Gripper for Agriculture Harvesting*, in: Proc. of the 25th Annual Conference on Towards Autonomous Robotic Systems (TAROS'24), vol. 15052, LNCS, Springer, 2025, pp. 266–277, ISBN: 978-3-031-72062-8, DOI: [10.1007/978-3-031-72062-8_24](https://doi.org/10.1007/978-3-031-72062-8_24).
- [13] J. P. ESPEJEL FLORES, A. YILMAZ, L. A. SORIANO AVENDAÑO, G. CIELNIAK: *Comparative Analysis of Unity and Gazebo Simulators for Digital Twins of Robotic Tomato Harvesting Scenarios*, in: Proc. of the 25th Conference Towards Autonomous Robotic Systems (TAROS'24), vol. 15051, LNAI, Springer, 2025, pp. 15–27, ISBN: 978-3-031-72059-8, DOI: [10.1007/978-3-031-72059-8_2](https://doi.org/10.1007/978-3-031-72059-8_2).
- [14] R. M. FRENCH: *Catastrophic forgetting in connectionist networks*, Journal of Trends in Cognitive Sciences Vol. 3.Issue 4 (1999), pp. 128–135, DOI: [10.1016/s1364-6613\(99\)01294-2](https://doi.org/10.1016/s1364-6613(99)01294-2).
- [15] S. GOURAGUINE, M. QBADOU, M. RAFIK, M. RIAD, K. MANSOURI: *A new knowledge primitive of digits recognition for NAO robot using MNIST dataset and CNN algorithm for children's visual learning enhancement*, Journal of Information Technology Education: Research 22 (2023), pp. 389–408, DOI: [10.28945/5194](https://doi.org/10.28945/5194).
- [16] N. JAMIL, O. S. ALAWA, S. M. MANAR, S. A. ALAIDAROUS, A. S. ADAM, S. A. ELDEMERDASH, A. N. BELKACEM: *On Neurodevelopmental Disorder based on Brain Computer Interface for Enhancing the Learning Process*, in: Proc. of the 2023 IEEE Frontiers in Education Conference (FIE'23), vol. 1, Institute of Electrical and Electronics Engineers, 2023, pp. 1–9, DOI: [10.1109/FIE58773.2023.10343434](https://doi.org/10.1109/FIE58773.2023.10343434).
- [17] N. JAMIL, A. BELKACEM, A. LAKAS: *On enhancing students' cognitive abilities in online learning using brain activity and eye movements*, English, Jorunal of Education and Information Technologies Vol. 28.Issue 4 (Apr. 2023), pp. 4363–4397, ISSN: 1360-2357, DOI: [10.1007/s10639-022-11372-2](https://doi.org/10.1007/s10639-022-11372-2).
- [18] N. JAMIL, A. N. BELKACEM, S. OUHBI, C. GUGER: *Cognitive and Affective Brain-Computer Interfaces for Improving Learning Strategies and Enhancing Student Capabilities: A Systematic Literature Review*, IEEE Access 9 (2021), pp. 134122–134147, DOI: [10.1109/ACCESS.2021.3115263](https://doi.org/10.1109/ACCESS.2021.3115263).
- [19] KA, JAYASHEEL KUMAR, AKHTAR, MANNAAN, TALHA, C. MOHAMMED, VG, GOKUL: *Design of Hybrid Sprinkler: The IoT-Powered Robot for Watering Plants*, MATEC Web Conf. 393 (2024), p. 04004, DOI: [10.1051/mateconf/202439304004](https://doi.org/10.1051/mateconf/202439304004).
- [20] L. MADUREIRA, A. SOUSA, J. BRAGA, P. CALADO, P. DIAS, R. MARTINS, J. PINTO, J. SOUSA: *The light autonomous underwater vehicle: Evolutions and networking*, in: Proc. of 2013 MTS/IEEE OCEANS Conference - Bergen (OCEANS'13), vol. 1, Institute of Electrical and Electronics Engineers, 2013, pp. 1–6, DOI: [10.1109/OCEANS-Bergen.2013.6608189](https://doi.org/10.1109/OCEANS-Bergen.2013.6608189).
- [21] R. R. MURPHY: *“Competing” for a robotics education*, IEEE Robotics & Automation Magazine 8.2 (2001), pp. 44–55.
- [22] S. NARVEKAR, B. PENG, M. LEONETTI, J. SINAPOV, M. E. TAYLOR, P. STONE: *Curriculum learning for reinforcement learning domains: A framework and survey*, Journal of Machine Learning Research 21.181 (2020), pp. 1–50.
- [23] J. Q. PÉREZ, T. DARADOU MIS, J. M. M. PUIG: *Rediscovering the use of chatbots in education: A systematic literature review*, Computer Applications in Engineering Education 28.6 (2020), pp. 1549–1565.

- [24] A. PLATZER: *The Significance of Symbolic Logic for Scientific Education* (2024), DOI: [10.48550/ARXIV.2407.09959](https://doi.org/10.48550/ARXIV.2407.09959), arXiv: [2407.09959](https://arxiv.org/abs/2407.09959).
- [25] K. Y. ROZIER: *On Teaching Applied Formal Methods in Aerospace Engineering*, in: *Formal Methods Teaching*, Cham: Springer International Publishing, 2019, pp. 111–131, ISBN: 978-3-030-32441-4.
- [26] D. SETO, B. KROGH, L. SHA, A. CHUTINAN: *The Simplex architecture for safe online control system upgrades*, in: *Proc. of the 1998 American Control Conference. (ACC'98)*, vol. 6, IEEE, Institute of Electrical and Electronics Engineers, 1998, pp. 3504–3508, DOI: [10.1109/ACC.1998.703255](https://doi.org/10.1109/ACC.1998.703255).
- [27] S. SHIBI, S. ZAIDI: *STEM Approach to Enhance Robot-Human Interaction Through AI Large Language Models and Reinforcement Learning*, in: *2024 IEEE Integrated STEM Education Conference (ISEC)*, 2024, pp. 1–2, DOI: [10.1109/ISEC61299.2024.10665163](https://doi.org/10.1109/ISEC61299.2024.10665163).
- [28] SOFTBANK ROBOTICS: *Meet Pepper: The Robot Built for People*, <https://us.softbankrobotics.com/pepper>.
- [29] SOFTBANK ROBOTICS: *NAO: Personal Robot Teaching Assistant*, <https://us.softbankrobotics.com/nao>.
- [30] E. VÁZQUEZ-CANO, S. MENGUAL-ANDRÉS, E. LÓPEZ-MENESES: *Chatbot to improve learning punctuation in Spanish and to enhance open and flexible learning environments*, *International Journal of Educational Technology in Higher Education* 18 (2021), pp. 1–20.
- [31] S. H. VEMPRALA, R. BONATTI, A. F. C. BUCKER, A. KAPOOR: *ChatGPT for Robotics: Design Principles and Model Abilities*, *IEEE Access* 12 (2023), pp. 55682–55696, URL: <https://api.semanticscholar.org/CorpusID:259141622>.
- [32] S. WANG, F. WANG, Z. ZHU, J. WANG, T. TRAN, Z. DU: *Artificial intelligence in education: A systematic literature review*, *Expert Systems with Applications* 252 (2024), p. 124167, ISSN: 0957-4174, DOI: [10.1016/j.eswa.2024.124167](https://doi.org/10.1016/j.eswa.2024.124167), URL: <https://www.sciencedirect.com/science/article/pii/S0957417424010339>.
- [33] Z. WARREN, Z. ZHENG, A. SWANSON, E. BEKELE, L. ZHANG, J. A. CRITTENDON, A. WEITLAUF, N. SARKAR: *Can robotic interaction improve joint attention skills?*, *Journal of Autism and Developmental Disorders* 45 (11 2013), pp. 3726–3734, DOI: [10.1007/s10803-013-1918-4](https://doi.org/10.1007/s10803-013-1918-4).

Making the boss smarter: A journey from rule-based to learned behaviors

Réka Erdész, Ede Troll

Eszterházy Károly Catholic University
reka.erdesz@uni-eszterhazy.hu
troll.ede@uni-eszterhazy.hu

Abstract. This paper investigates the application of reinforcement learning to non-player character (NPC) behavior design in a roguelike video game, with the goal of creating more engaging and less predictable opponents. Two training approaches were compared using the Unity ML-Agents framework: Agent A, trained exclusively through self-play, and Agent B, trained initially against human players before switching to self-play. Performance was evaluated using quantitative metrics such as policy loss, value loss, entropy, and ELO ratings, alongside qualitative feedback from semi-professional players. While Agent B achieved faster convergence and higher ELO scores, player feedback indicated a preference for Agent A due to its unpredictability, balanced tactics, and lower frustration levels. The results highlight the need to balance technical optimization with player experience, and suggest that hybrid training strategies may yield the most compelling adversaries in future game AI design.

Keywords: reinforcement learning, game AI, Unity ML-Agents, player experience, NPC behavior

AMS Subject Classification: 68T05, 68T07, 91A35

1. Introduction

This study examines the integration of reinforcement learning techniques into the design of non-player character (NPC) behaviors within a roguelike video game environment. The research aims to address limitations inherent in traditional rule-based artificial intelligence (AI) approaches, which often lead to predictable and

monotonous adversary behavior, thereby reducing gameplay challenge and engagement [11]. Notable successes in reinforcement learning, such as mastering the game of Go through self-play [9], highlight the potential of these techniques to create adaptive and challenging opponents.

The initial implementation employed a rule-based AI [2], utilizing the A* path-finding algorithm for navigation and a behavior tree [5] for the final boss. Although the behavior tree introduced conditional actions – such as retreating when the agent’s health was low or becoming aggressive when the player was weakened – the decision-making process remained deterministic and easily exploitable by experienced players.

To overcome these constraints, We implemented two alternative AI training methodologies using the Unity Machine Learning Agents (ML-Agents) framework [4]. Both agents were trained via reinforcement learning [10], enabling them to improve their decision-making through interaction with the game environment. Agent A was trained entirely through self-play, iteratively competing against its previous policy versions. Agent B underwent an initial training phase of 30,000 steps against human players before switching to self-play. This dual approach allowed us to compare the benefits of purely autonomous learning against a hybrid method incorporating human-guided exploration.

The primary objective of this research is to evaluate which training paradigm yields a more challenging, engaging, and strategically capable opponent. Evaluation was conducted through both quantitative performance metrics and qualitative human player feedback, with the ultimate goal of identifying design practices that enhance player experience while maintaining balanced gameplay difficulty.

1.1. Game concept and the original enemy AI

The game [7] is a room-based rogue-like video game. In each room, the player has to defeat randomly generated enemies, fighting their way to the final boss. Initially, enemies were operating on a rule-based system; their movements were based on the A* algorithm. While the smaller enemies only had a few attack types, the final boss had a so-called *Behavior Tree* [5], which resulted in a better experience. The behavior tree gave the boss some human-like behavior based on the health percentage of its own and its opponent’s – the player’s health. It retreated when it had a low health percentage, and it became aggressive when the player had low health – but it was still predictable. The agent’s actions were defined by specific rules, making the enemy’s responses monotonous and easy to anticipate. This predictability reduced the overall gameplay challenge.

1.2. Teaching process of the enemy

For the research [3], We used the Unity ML-Agents toolkit for teaching purposes. This is an open-source toolkit developed by Unity Technologies in 2020. The core concept involves an Agent that learns through reinforcement learning. Unity provides a plug-in package that includes the necessary code libraries and connects to

the virtual environment. To get started, We had to create a virtual environment and, like with all machine learning projects, We needed to install TensorBoard and PyTorch using pip. Additionally, We installed ML-Agents, which requires NumPy and TensorFlow.

1.2.1. ML-Agents

The ML-Agents framework is centered around reinforcement learning, where an agent learns to make decisions in various situations based on feedback from its environment, known as rewards. This learning process does not rely on examples from teachers; instead, it is driven by self-trial and error. The feedback received from the environment aids in developing an optimal strategy for the agent.

Formally, this process is described by a Markov Decision Process (MDP) [6], whose components are:

$$\mathcal{M} = \langle S, A, P, R, \gamma \rangle$$

An MDP is a mathematical framework used to model decision-making in environments where outcomes are partly random and partly under the control of the agent.

State space (S) The state space defines all observable features available to the agent during decision-making. In this study, the agent’s state vector consisted of positional, health, status, timing, and event-related variables, as summarized in Table 1.

Table 1. Summary of state space components.

Category	Description
Positional data	Distance from the player; direction of the player relative to the agent; agent’s facing direction.
Health information	Current health points of the agent; current health points of the player.
Status flags	Attack availability (boolean flag preventing overlapping attacks); whether the agent is currently attacking.
Timing	Time elapsed since the last attack.
Special events	Whether the player is affected by the “black hole” status (immobilized).

The first experiment We conducted was a test on reinforcement learning with a hardcoded AI opponent. We trained an Agent against this opponent, and this is where We first initialized the reward system. Initially, the rewards were set too high, which made the agent’s policy unstable. To address this, We normalized the large integers to a $0 \dots x$ interval, resulting in more stable learning.

Although it is important from a teaching perspective to know how much reward the agent received and why, the essence of the research lies in the methodology, which can be particularly relevant in roguelike games. The reward system will differ for each developer’s model; as an extension, we aim to apply this methodology to other genres and multiple games.

Action space (A) Table 2 lists the discrete actions available to the agent at each decision step.

Table 2. Summary of action space components.

Action	Description
Move forward/backward	Adjusts the agent’s position along the facing direction.
Strafe left/right	Moves perpendicular to the current facing direction.
Rotate left/right	Changes the facing direction of the agent.
Basic attack	Executes the standard melee or ranged attack, depending on the agent’s class.
Special abilities	Uses special skills such as “black hole” or other area-of-effect attacks.
Retreat/Heal	Moves away from the player and recovers health if possible.
Idle	Performs no action for a single decision step.

Reward system (R) The agent received rewards for winning, using varied attacks (to discourage attack spamming), surviving with low health (encouraging prolonged combat), and performing tactical retreats when low on HP. We initially gave a reward for movement, but zig-zag movement developed, so we removed that reward. This led to the agent learning to move strategically on its own. General penalties were applied for taking damage, standing still, and missing hits.

Table 3 summarizes the conditions for rewards and penalties.

One of the key lessons learned during training was that relying solely on the reward system was insufficient. In many cases, manual adjustments to the weights of rewards and penalties were necessary to improve learning stability or to address unwanted behaviors.

The main part of the agent’s reward system was the ability system. First, a base reward was introduced that the agent received for every reward event, based on the damage ratio:

$$D = \frac{\text{damage}}{\text{playerCurrentHP}}$$

Table 3. Reward and penalty conditions.

Event	Type
Winning a match	Reward
Successful attack hit	Reward
Varied attack usage	Reward
Survival with low health	Reward
Tactical retreat when low HP	Reward
Taking damage	Penalty
Standing still (no action)	Penalty
Missed attack	Penalty
Reward for movement (removed due to zig-zag behavior)	Removed reward

1.2.2. PPO

PPO – Proximal Policy Optimization [8], is a popular reinforcement learning algorithm often used in game development because it stably improves AI decisions without sudden big jumps in learning. This is especially important when the opponent is not deterministic, such as a human player, so the AI learns and adapts gradually. PPO constrains policy changes so that the ratio of new to old policies can only move within a certain range, thus facilitating the learning of complex strategies and the retention of prior knowledge.

We chose it for the project because it handles complex decision situations such as healing or tactical switching during combat well, and because it is natively supported by Unity ML-Agents, it is easy to integrate.

1.2.3. Agent A

Agent A was trained exclusively through self-play, meaning it fought against its previous versions, a technique inspired by successful applications like AlphaGo [9], which leveraged self-play to master complex strategic games. In the YAML configuration, we gradually increased the batch size (512), buffer size (5120), and entropy value (0.02), and linearly decreased the learning rate to ensure stable and continuous improvement. The design of the reward system was guided by principles of player-centric AI design [11], aiming to encourage varied and engaging behaviors rather than repetitive actions.

The self-play settings *included* `save_steps = 20000`, `team_change = 150000`, `swap_steps = 5000`, which ensured that opponents were periodically refreshed. This prevented the agent from getting stuck in repetitive patterns and allowed it to learn against a variety of opponents. The selected model was chosen from the run that produced the lowest policy and value loss values.

1.2.4. Agent B

Agent B took its first 30 000 steps fighting against human players, then switched to self-play learning. With player guidance, the AI learned relevant patterns early on, its behavior stabilized more quickly, and it mastered tactical elements better – such as when to heal or avoid combat. With Agent B, we encountered the well-known problem in reinforcement learning: reward hacking. I mentioned this phenomenon in 1.2.1 that the agent found a loophole in the reward system with movement rewards. Our next research is based on this experience.

Hyperparameters

For reproducibility, we detail the complete PPO setup for both agent in Table 4.

Table 4. PPO hyperparameters used in training.

Parameter	Value
Learning rate	3×10^{-4}
Batch size	512
Buffer size	5120
Gamma (γ)	0.99
GAE λ	0.95
Entropy coefficient (beta)	0.001
Clip range (epsilon)	0.3
Number of epochs	5
Number of environments	1
Hidden units	128
Number of layers	2
Time horizon	64
Memory sequence length	64
Memory size	128
Seed	-1 (random)

Evaluation Protocol (Appendix)

The evaluation protocol was based on the self-play configuration in the YAML files. The details are as follows:

- **Opponent pool:** maintained as a sliding window of the last 10 models (defined by `window: 10`).
- **Checkpoint saving:** new models were added to the pool every 20,000 steps (`save_steps: 20000`).

- **Team change:** enforced after 150,000 steps (`team_change: 150000`).
- **Opponent swap frequency:** opponents were rotated every 5,000 steps (`swap_steps: 5000`).
- **Latest model ratio:** 50% of matches were played against the most recent model (`play_against_latest_model_ratio: 0.5`).
- **Initial ELO:** all agents started at 1200 (`initial_elo: 1200.0`).

1.3. Measurements

To evaluate the agents with metrics, we used TensorBoard and evaluated the following metrics to assess AI performance:

- **Policy Loss:** how stable the AI strategy is.
- **Value Loss:** how well it can predict future rewards, see Figure 1, 2.
- **Entropy:** the variability of decisions can be seen in Figure 3.
- **ELO:** the relative strength between different AI models, see Figure 4.

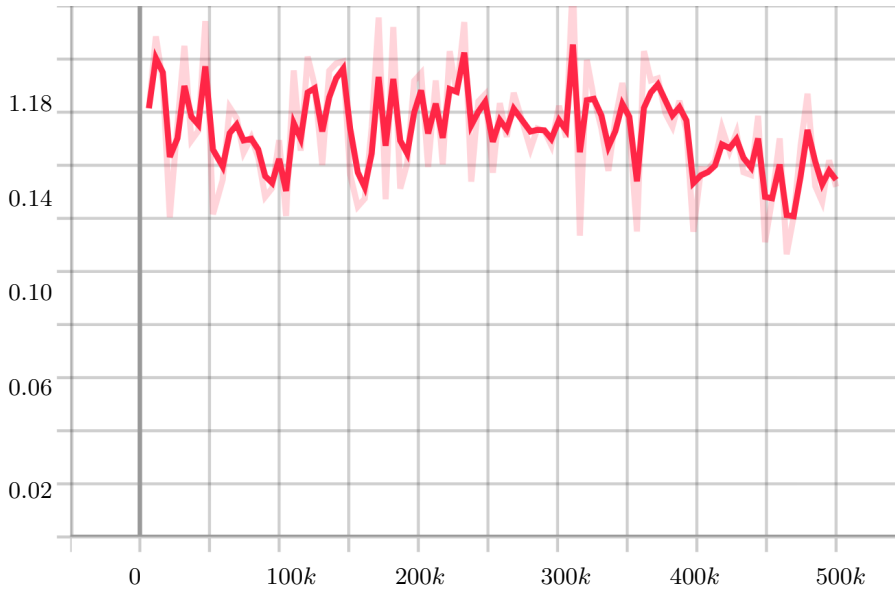


Figure 1. *Agent A's* TensorBoard scalar.

The agent with initial intelligence, trained against the player, showed more stable and rapid progress. Its Policy Loss quickly converged to around 0.04, see

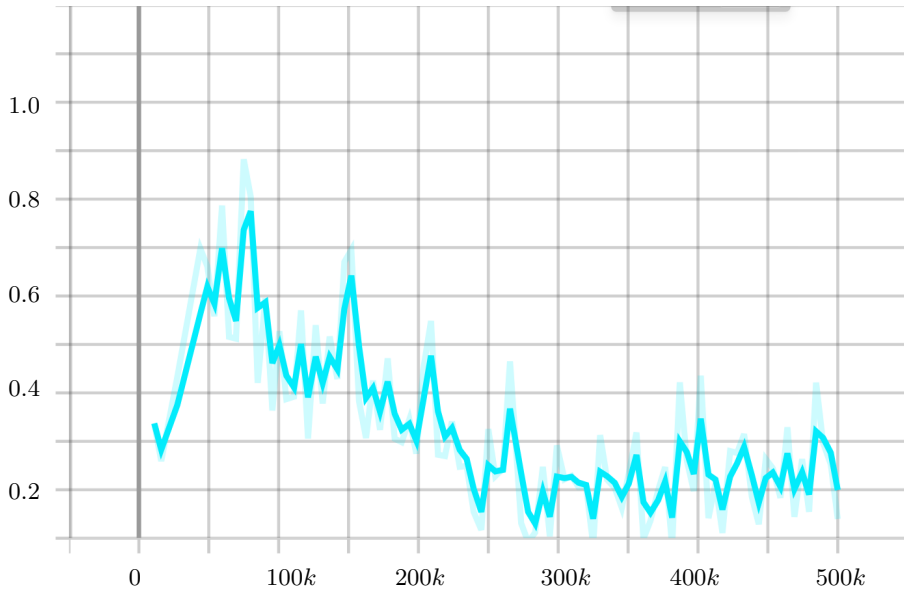


Figure 2. *Agent B's* TensorBoard scalar.

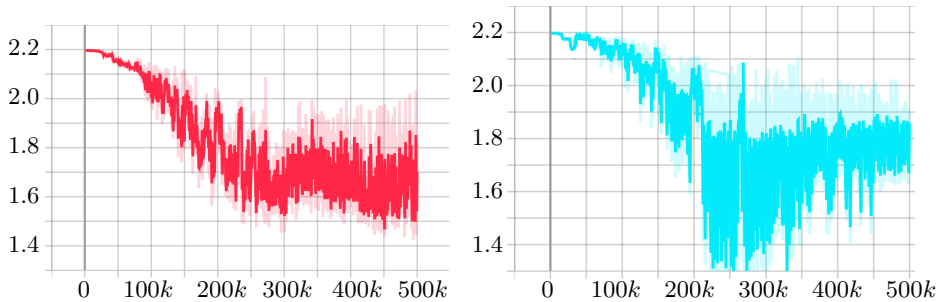


Figure 3. On the left diagram, *Agent A's* TensorBoard scalar of entropy can be seen, while *Agent B* is on the right.

Figure 2, while the self-learning agent began to improve only after several hundred thousand steps, see Figure 1, exhibiting significant fluctuations. Both agents demonstrated a gradual decrease in Value Loss, indicating effective reward prediction.

Entropy values stabilized between 1.7, and 1.8, see Figure 3 for both agents, reflecting their ability to explore without becoming entirely random. However, the ELO curves highlighted differences: the agent with initial intelligence started with a higher ELO and maintained competitiveness, while the self-learning agent's ELO gradually declined, likely due to overly narrow strategies.

In summary, the agent that began with initial knowledge appears to be a more formidable enemy, based on the indicators above.

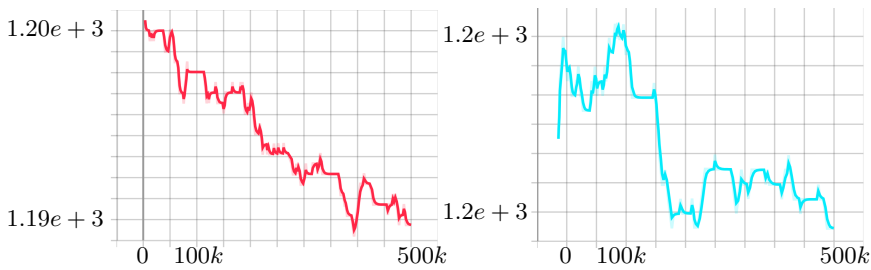


Figure 4. On the left diagram, *Agent A*'s TensorBoard scalar can be seen, while *Agent B* is on the right.

1.4. Results

To evaluate the behavior of the two AIs, we asked ten semi-professional players to try out the two versions. Participants completed a 22-question questionnaire that assessed gameplay experience, frustration, strategic depth, and predictability.

1.4.1. Gaming experience and preference

Agent A received an average rating of 4 out of 5, compared to Agent B's 2.6 out of 5 (Q4-Q5). Additionally, frustration levels were significantly different: Agent A scored 2.4 out of 5, while Agent B scored 4 out of 5 (Q15-Q16), suggesting more negative experiences with Agent B, see Figure 5.

1.4.2. Evaluation of strategy and tactics

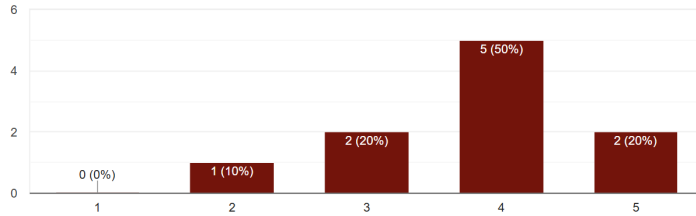
It is interesting to note that, despite Agent A being more popular, 70% of players (7 out of 10) believed that Agent B had more advanced strategic skills (Q11). However, Agent A was perceived as more unpredictable (Q8), while Agent B's behavior was often predictable. In some cases, this predictability led to a negative gaming experience, particularly due to the overuse of the black hole skill, which received criticism from several respondents (Q13), see Figure 6.

1.4.3. Realism of AI Behavior

Participants assessed the realism of each agent's behavior to determine how closely they resembled a human opponent. According to Q18, the majority (6/10) perceived neither agent as particularly human-like, with 2/10 favoring Agent A and 2/10 favoring Agent B. Quantitative ratings further revealed that Agent B was perceived as slightly more realistic, with an average score of 3.1/5 (Q20) compared to Agent A's 2.6/5 (Q19). This difference likely stems from Agent B's more advanced strategic patterns, as evidenced by 90% of participants (Q11) noting its superior strategies. However, the relatively low realism scores for both agents suggest that further refinements are needed to enhance the perception of human-like decision-making, a critical factor for immersive gameplay in roguelike games.

Q4. Rate your level of enjoyment while playing against Agent A (1 = Not enjoyable at all, 5 = Extremely enjoyable)

10 responses



Q5. Rate your level of enjoyment while playing against Agent B (1 = Not enjoyable at all, 5 = Extremely enjoyable)

10 responses

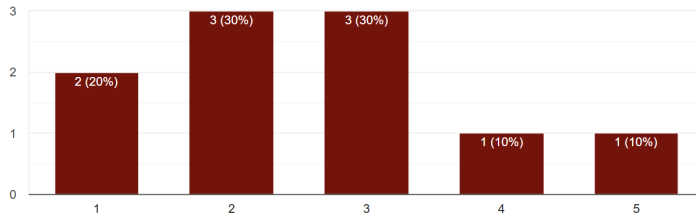


Figure 5. On the top diagram, *Agent A*'s enjoyment results can be seen, while *Agent B* is on the bottom.

Q11. Which AI seemed to have better strategies against the player? (If you noticed any)

10 responses

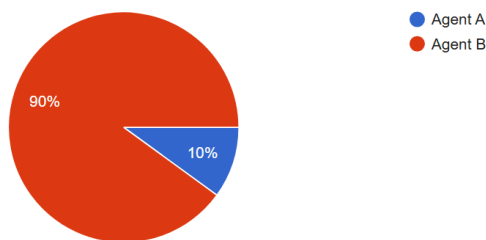


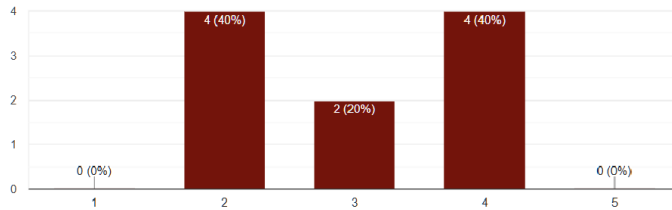
Figure 6. 9 out of ten found Agent B to have better strategies, even though they found him stronger.

1.4.4. Surprise factor in AI tactics

The extent to which the agents surprised players with their tactics was evaluated through Q21. On average, participants rated the surprise factor at 2.4/5, with 4/10 reporting rare surprises, 3/10 noting occasional surprises, and 2/10 indicating frequent surprises. Despite Agent A being perceived as less predictable (Q8), the

Q9. How unpredictable was Agent A's behavior? (Higher = less predictable, more interesting)

10 responses



Q10. How unpredictable was Agent B's behavior? (Higher = less predictable, more interesting)

10 responses

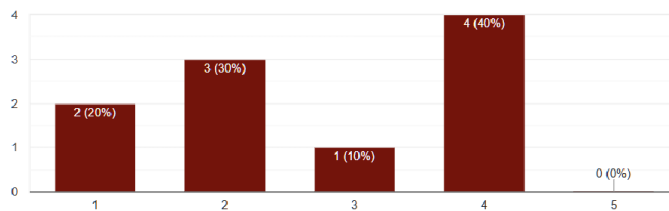


Figure 7. On the top diagram, *Agent A's* enjoyment results can be seen, while *Agent B* is on the bottom.

moderate surprise ratings suggest that its unpredictability did not consistently translate into novel or innovative tactics. This distinction highlights a potential area for improvement in designing AI behaviors that not only vary in response but also introduce genuinely unexpected strategies to enhance the challenge and replayability of the game.

1.4.5. Impact of reward hacking on perceived unpredictability

Player feedback highlighted a significant issue with Agent B's overuse of the black hole ability, with 60% of participants (Q13) noting that it led to frustrating and repetitive gameplay. One respondent explicitly suggested reward hacking, observing that Agent B used the ability excessively despite designed constraints (e.g., limiting void rifts to two per 10 seconds). This behavior likely contributed to Agent B's lower unpredictability score of 3/5 (Q10) which can be seen in Figure 7 compared to Agent A's 3.2/5, despite its more advanced strategies (Q11). The paradox of reward hacking reducing perceived unpredictability underscores the importance of carefully calibrating the reward system to prevent exploitative behaviors that undermine gameplay variety and player satisfaction.

Overall scores

- **Average gaming experience:** Agent A – 4/5, Agent B – 2.6/5.

- **Frustration:** Agent A – 2.4/5, Agent B – 4/5.
- **Strategic perception:** 70% said Agent B used more advanced strategies, but Agent A was considered more unpredictable.
- **Boss preference:** 7 out of 10 players chose Agent A as their final boss.

Several players criticized Agent B for overusing the “black hole” ability, which made gameplay predictable and sometimes frustrating. In contrast, Agent A had a more varied use of abilities and balanced behavior.

1.4.6. Ultimate preference – Which AI should be the arch enemy?

Most respondents (7 out of 10) selected Agent A as their preferred final boss enemy (Q22), see Figure 8. Agent A was noted for its diverse skill usage, balanced behavior, and lower frustration scores. In contrast, Agent B received more criticism due to its aggressive and often overly dominant use of abilities, particularly the black hole.

Q22. If one of the AIs had to be used as the final boss in the game, which would you prefer?

10 responses

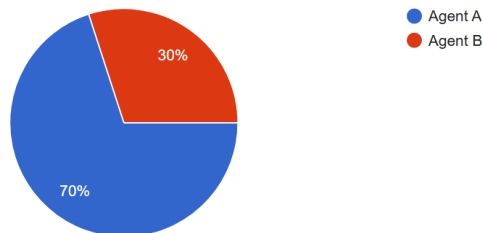


Figure 8. 7 out of 10 people came to the conclusion after answering all the questions and testing both agents that Agent A is their ultimate preference for a game boss.

2. Conclusion

Quantitative analysis using TensorBoard metrics demonstrated that Agent B, trained with initial human guidance, achieved faster convergence, higher ELO ratings, and more stable policy optimization compared to Agent A, which relied solely on self-play. However, qualitative feedback from ten semi-professional players revealed a preference for Agent A, attributed to its greater unpredictability, more varied skill usage, and lower frustration levels (mean: 2.4/5 vs. Agent B’s 4/5). Notably, Agent B’s overuse of the black hole ability, reported by 60% of participants, not only increased frustration but also paradoxically reduced its perceived unpredictability.

(3/5 vs. Agent A's 3.2/5), likely due to reward hacking exploiting the reward system design.

Furthermore, neither agent was consistently perceived as human-like, with Agent B rated slightly higher in realism (3.1/5 vs. 2.6/5), suggesting that advanced strategies alone do not suffice to create the illusion of a human opponent. Additionally, the moderate surprise factor (mean: 2.4/5) indicated that unpredictability did not always translate into novel or innovative tactics, potentially limiting long-term player engagement. These findings underscore the necessity of balancing technical optimization with player experience objectives, prioritizing fairness, tactical variety, and the capacity to surprise players for an engaging gameplay experience.

Future work will explore hybrid training methodologies combining autonomous self-play with periodic human-guided refinement to enhance both strategic competence and perceived realism. Additionally, addressing reward hacking through refined reward structures, such as temporal constraints on ability usage or diversified penalties, will be critical to prevent exploitative behaviors. Approaches like Group Relative Policy Optimization (GRPO)[1] and multi-agent adaptive systems may further ensure that AI opponents remain strategically robust while delivering enjoyable and immersive gameplay experiences.

References

- [1] Y. CHENG, Y. LI, Q. LIU, T. ZHANG, Y. ZHAO, B. JIANG, Y. ZHU, S. LU, W. XIONG, J. HE, ET AL.: *DeepSeekMath: Teaching Large Language Models to Reason through Latent Programs with Group Relative Policy Optimization*, arXiv preprint arXiv:2402.03300 (2024), DOI: [10.48550/arXiv.2402.03300](https://doi.org/10.48550/arXiv.2402.03300), URL: <https://arxiv.org/abs/2402.03300>.
- [2] J. DORAN, I. PARBERRY: *Integrating Rule-Based AI Tools into Mainstream Game Development*, International Journal of Computer Games Technology 2024 (2024), DOI: [10.1007/978-3-319-99906-7_23](https://doi.org/10.1007/978-3-319-99906-7_23).
- [3] R. ERDÉSZ: *ARM509_Thesis*, https://github.com/frake92/ARM509_Thesis, Accessed: 2025-08-10, 2025.
- [4] A. JULIANI, V.-P. BERGES, J. TANG, R. CARR, J. TOGELIUS, D. LANGE: *Unity: A General Platform for Intelligent Agents*, in: vol. 33, 01, 2019, DOI: [10.48550/arXiv.1809.02627](https://doi.org/10.48550/arXiv.1809.02627).
- [5] R. MARCOTTE, H. J. HAMILTON: *Behavior Trees for Modelling Artificial Intelligence in Games: A Tutorial*, The Computer Games Journal 6.3 (Sept. 2017), pp. 171–184, ISSN: 2052-773X, DOI: [10.1007/s40869-017-0040-9](https://doi.org/10.1007/s40869-017-0040-9).
- [6] M. L. PUTERMAN: *Chapter 8 Markov decision processes*, in: Stochastic Models, vol. 2, Handbooks in Operations Research and Management Science, Elsevier, 1990, pp. 331–434, DOI: [10.1016/S0927-0507\(05\)80172-0](https://doi.org/10.1016/S0927-0507(05)80172-0), URL: <https://www.sciencedirect.com/science/article/pii/S0927050705801720>.
- [7] *Quantum Crusade*, <https://apestudiosinc.itch.io/quantum-crusade>, Accessed: 2025-08-10.
- [8] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, O. KLIMOV: *Proximal Policy Optimization Algorithms*, arXiv preprint arXiv:1707.06347 (2017), DOI: [10.48550/arXiv.1707.06347](https://doi.org/10.48550/arXiv.1707.06347).
- [9] D. SILVER, J. SCHRITTWIESER, K. SIMONYAN, I. ANTONOGLOU, A. HUANG, A. GUEZ, T. HUBERT, L. BAKER, M. LAI, A. BOLTON, ET AL.: *Mastering the game of Go without human knowledge*, Nature 550 (2017), DOI: [10.1038/nature24270](https://doi.org/10.1038/nature24270).

- [10] R. S. SUTTON, A. G. BARTO: *Reinforcement Learning: An Introduction*, 2nd, MIT Press, 2018, URL: <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>.
- [11] G. N. YANNAKAKIS, J. TOGELIUS: *Artificial Intelligence and Games*, Springer, 2018, DOI: [10.1007/978-3-319-63519-4](https://doi.org/10.1007/978-3-319-63519-4).

Multi-objective genetic and memetic algorithms in flexible flowshop scheduling

Levente Áron Fazekas^a, Károly Nehéz^b

^aInstitute of Information Technology, University of Miskolc
levente.fazekas@uni-miskolc.hu
károly.nehez@uni-miskolc.hu

Abstract. Flowshop scheduling problems are classic examples of scheduling where the objective is to minimize makespan – the total manufacturing time. Only the processing times required for each operation are considered. Since the general flowshop problem is NP-hard, multiple heuristic and metaheuristic approaches have emerged over the years. Most practical applications, however, require a much more nuanced approach: multiple – sometimes contradictory – objectives must be considered simultaneously alongside a plethora of additional constraints. Flexible flowshop problems are an abstraction of classic flowshops, where each stage can consist of multiple parallel machines, referred to as work centres. Commonly, models also have to consider a broader range of manufacturing restrictions and variables, such as setup times, machine eligibility restrictions, and due dates. This study aims to demonstrate the application of genetic and memetic metaheuristic algorithms on the $FFc \mid s_{i,j,k}, d_j, M_j \mid C_{max}, T_{max}, \sum_j T_j, \sum_j U_j$ flexible flowshop scheduling problem. It also outlines a dynamic decoding method for permutation or random key representations to alleviate controllability and tightness problems during genotype-phenotype conversion. Common genetic crossover and mutation operations are showcased alongside the simulated annealing local search algorithm to form memetic algorithms. To handle multiple objectives, a modified version of the relative distance method is employed. The findings are demonstrated via the Taillard benchmark set.

Keywords: genetic algorithm, memetic algorithm, simulated annealing, multi-objective scheduling, flowshop scheduling.

AMS Subject Classification: 90B35, 90C23

1. Introduction

Flowshop scheduling problems are classic examples of constrained multi-resource and multi-operation scheduling [12]. Classically, the objective is to minimize make-

span – the total manufacturing time – where only the processing times required for each operation are considered. Since the general flowshop problem is NP-hard, multiple heuristic and metaheuristic approaches have emerged over the years. Most practical applications, however, require a much more nuanced approach: multiple – sometimes contradictory – objectives must be considered simultaneously alongside a plethora of additional constraints. Flexible flowshop problems [10, 20] are an abstraction of classic flowshops, where each stage can consist of multiple parallel machines, referred to as work centres. Commonly, models also have to consider a broader range of manufacturing restrictions and variables, such as setup times, machine eligibility restrictions, and due dates [21, 23]. This study aims to demonstrate the application of genetic and memetic metaheuristic algorithms on the $FFc \mid s_{i,j,k}, d_j, M_j \mid C_{max}, T_{max}, \sum_j T_j, \sum_j U_j$ flexible flowshop scheduling problem. It also outlines a dynamic decoding method [29] for permutation or random key representations [26] to alleviate controllability and tightness problems during genotype-phenotype conversion. Common genetic crossover and mutation operations are showcased alongside the simulated annealing local search algorithm to form memetic algorithms. To handle multiple objectives, a modified version of the relative distance method [14, 15] is employed as opposed to common weighted sum or ε -constraint methods [2] or non-dominating sorting genetic algorithms [5, 19]. The findings are demonstrated via the Taillard benchmark set [24], where the original problem has been transformed to accommodate the denoted problem.

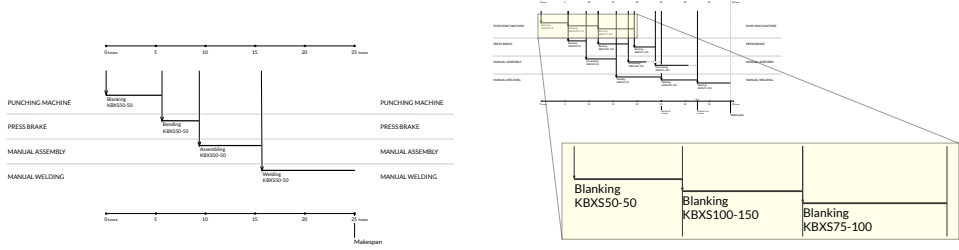
Most flexible flowshop studies choose to focus on throughput-related performance indicators, such as minimizing makespan or flow time. In particular cases, throughput may not be the most important or the only objective. In make-to-order manufacturing, a late order implies a penalty in the form of loss of goodwill, and the magnitude of the penalty depends on the tardiness of the delivery [20]. In many circumstances, managing on-time delivery has significance alongside improving the system's throughput. Optimizing due-date-related schedule metrics, such as the number of tardy orders, total tardiness, and maximum lateness, is crucial for manufacturing firms.

In the literature, algorithms for solving the flexible flowshop problem can be categorized into exact and heuristic approaches. Exact methods, including mathematical programming and branch and bound, create optimal solutions. Due to the lack of efficient lower bounds, the branch and bound approach is limited to simple shop configurations. Exact methods require a long time for solving large instances. Both facts limit the practical application of these methods. A more practical method is to search for a quasi-optimal solution in a reasonable amount of time. For this reason, the trend is to solve flexible flowshop problems using heuristics, especially metaheuristics.

2. Problem formulation

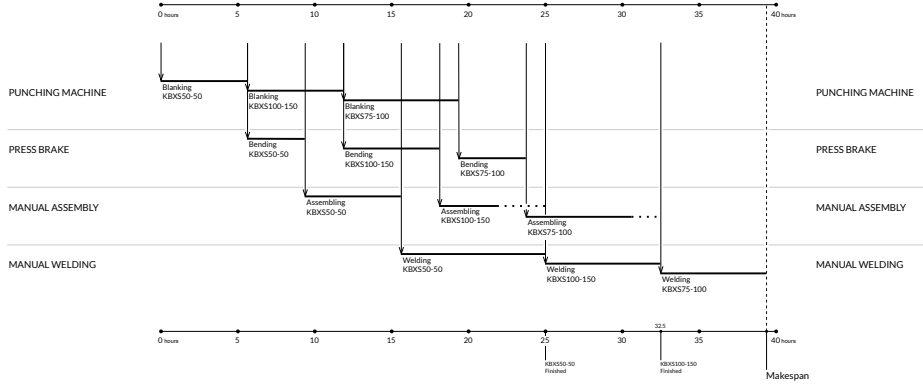
A scheduling problem can be described by a triplet $\alpha|\beta|\gamma$ notation [23]. The α field describes a machine environment and usually contains just one entry. The β field

details the processing characteristics and constraints of operations. This β field may contain multiple, single, or no entries. The γ field describes the performance metric or metrics to be minimized.



(a) An example of the first job contingency in the classic flowshop problem.

(b) An example of first machine contingency in the classic flowshop problem.



(c) An example of the flow-shop problem.

Figure 1. A classic flowshop problem.

Consider a flowshop F environment with n jobs and m machines: $p_{j,i}$ represent the processing times. $i \in \{1, 2, \dots, m\}$ is the i th machine in the production line, $S_{j,i}$ is the starting time of job j on machine i , $C_{j,i}$ is the completion time of job j on machine i . A resource can only process one job at a time; therefore, the start time of the next job must be equal to or greater than the completion time of its predecessor on the same resource. A job can only be present on one resource at any given time, meaning that the start time of the same job on the next machine must be greater than or equal to the completion time of its predecessor operation.

$$\begin{aligned}
 C_{j,i} &\leq S_{j+1,i} \\
 C_{j,i} &\geq C_{j,i+1} \\
 C_{1,i} &= \sum_{k=1}^i p_{1,k}
 \end{aligned}$$

$$C_{j,1} = \sum_{l=1}^j p_{j,l}$$

$$C_{j,i} = \max(C_{i-1,j}, C_{i,j-1}) + p_{i,j}$$

The first scheduled job does not have to wait for other jobs and is available from the start $S_{1,1} = 0$. The completion time of the first job on the current machine is always the sum of its previous operations on the preceding machine in the chain and its processing time on the current machine. Figure 1a shows how the first scheduled job has only one contingency - its operations on previous machines. Jobs on the first machine are only contingent on jobs on the same resource. Therefore, the completion time of the job on the first machine is the sum of previously scheduled jobs plus its own and $S_{i,1} = C_{i-1,1}$. Figure 1b illustrates that the first machine has only one contingency, and operations can follow without delay. Considering subsequent jobs on subsequent machines ($i, j > 1$), the completion times are contingent on the same job on previous machines and previously scheduled jobs on the previous machines in the chain. Figure 1c illustrates the solution for a classic flowshop problem using a Gantt chart.

The classic flowshop problem aims to minimize the completion time of the last job called the makespan. Therefore, the aim is to minimize the completion time of the last scheduled machine on the last machine in the manufacturing line:

$$C_{max} = C_{n,m} \rightarrow \min$$

2.1. A flexible flowshop example

A flexible flowshop is a generalization of the classic flowshop (Fm) and parallel machine (Pm) environments. Instead of m machines in series, there are c stages with several identical machines in parallel. Each job must be processed first at stage 1, then stage 2, and so on. A stage functions as a bank of parallel machines; at each stage, job j requires processing on only one machine, and any machine can do it. The queues between the various stages may or may not operate according to the *First Come First Served (FCFS)* principle. In literature, flexible flowshops have also been referred to as hybrid and multiprocessor flowshops. The following flexible flowshop problem is presented as an example:

$$FFc \mid s_{i,j,k}, d_j, M_j \mid C_{max}, T_{max}, \sum_j T_j, \sum_j U_j$$

In this paper, we present a problem with identical parallel machines at each stage (FFc), machine eligibility constraints (M_j), sequence-dependent setup times ($s_{i,j,k}$), due dates (d_j), and multiple objective functions. Machine eligibility indicates that not all machines can process any job in a stage due to certain limitations – this characteristic is significant in the modern industry but rarely considered by the literature [23].

For example, a four-stage flexible flowshop, a sheet metal manufacturing environment, is shown. The system consists of four stages: blanking, bending, welding, and assembling. Metal sheets enter the blanking stage, cutting the raw material into two-dimensional parts with a laser cutting machine or punching press. Then, the parts are transferred to the bending stage to be bent into specific three-dimensional parts. After bending, the welding and assembling stages take the parts to a completed - final product - state. In the cutting stage, a laser cutting machine may not be able to be used for all types of materials. Bending requires specific tool sets and work ranges, which may rule out specific machines for a particular part. Figure 2b illustrates such a four-stage flexible flowshop environment and a possible path in the system as opposed to the classic flowshop example illustrated in Figure 2a.

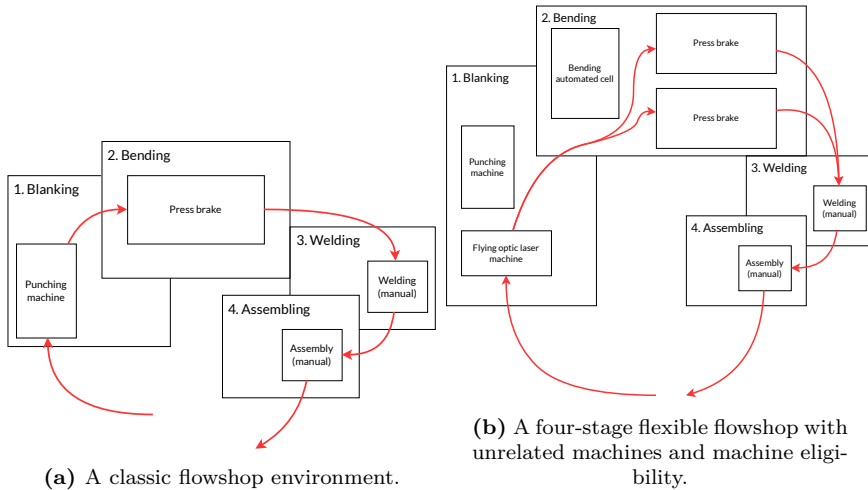


Figure 2. Comparison of Fm and FFc models.

3. Encoding and decoding methods

Encoding is a representation of a solution by a vector of values representing key decision variables on which an algorithm operates. This vector is often called a genotype or chromosome in case of genetic algorithms. Most generally, schedules are represented by each operation's start and finish times on every corresponding machine. This view allows for an infinite solution space. Since scheduling often involves minimizing makespan, flowtime, and lateness, all operations are commonly started as early as possible. This goal makes the schedule a semi-active schedule [20], in which no operation can be completed earlier without change the processing order. In such schedules, the decision variables are reduced to the machine assignment of each operation and the sequence of operations on each machine. A large encoding scheme in a large-scale solution may result in inefficient searching. Urlings

et al. [26] studied different encoding schemes in genetic algorithms and deduced that more detailed encodings result in worse scheduling performance. For this reason, most use a permutation encoding scheme, where a solution is represented as a job order $s \in \mathcal{S}_n$. Alternatively, random keys can replace direct permutations by mapping real valued vectors $s \in [0, 1]^n$ to job orders via sorting.

Decoding derives a schedule (phenotype) from the encoded solution (genotype). Permutation encoding does not contain all necessary decision variables when constructing a flexible flowshop schedule. These missing variables, like machine selection, are determined by heuristics during decoding; for this reason, decoding methods are crucial to solution quality. The most adopted method is List Scheduling (LS), where jobs are executed in the order given in the encoding on the first stage and using the first-come-first-served (FCFS) principle on all subsequent ones. With List Scheduling, the original ordering's influence on the schedule is diminished by the FCFS rule through the various stages. This limited influence is known as the *controllability problem*. Another widely used method is Permutation Scheduling (PS), as adopted by Ruiz and Stützle [22]. As opposed to List Scheduling, Permutation Scheduling keeps the initial, global ordering for all stages. This method improves control, making it easier to handle urgent jobs, but it can cause idle time when stages desynchronize. This inefficiency is called the *tightness problem*.

Despite their broad application, permutation and list scheduling both have obvious drawbacks. In scheduling, one may want to handle urgent jobs without the delay caused by synchronization. For this reason, Chunlong et al. [29] utilized permutation encoding and *Dynamic Scheduling* with *first available* machine selection to minimize the total tardiness $\sum_j T_j$ while maintaining tightness and controllability. In Dynamic Scheduling, both the completion time on the previous machine $C_{j,i-1}$ and the global order s are used. When a machine finishes a job, it chooses from jobs available at that time, but in the order given by the initial order s . This method combines List Scheduling and Permutation Scheduling by modifying the machine's buffer into a priority-queue.

Table 1. A 2-stage scheduling problem.

Job	Stage 1		Stage 2		Due
	Eligibility	Processing time	Eligibility	Processing time	
1	$\{M_{1,1}, M_{1,2}\}$	2	$\{M_{2,1}\}$	4	9
2	$\{M_{1,1}, M_{1,2}\}$	2	$\{M_{2,1}\}$	3	12
3	$\{M_{1,1}, M_{1,2}\}$	5	$\{M_{2,1}\}$	2	8

Table 1 presents a two-stage flowshop problem as an illustrative example. Suppose the solution is given by the Earliest Due Date (EDD) heuristic, resulting in the job sequence $s = \{3, 1, 2\}$. List Scheduling is depicted in Figure 3a. Due to differing completion times in stage 1, the job order in stage 2 changes to $s = \{1, 2, 3\}$. The resulting performance indicators are: makespan $C_{\max} = 11$, total tardiness $\sum_j T_j = 3$, maximum tardiness $T_{\max} = 3$, and number of late jobs $\sum_j U_j = 1$ (Job 3). Figure 3b shows the outcome of applying permutation-based scheduling.

To maintain the original sequence s across both stages, the start times of Jobs 1 and 2 in stage 2 are delayed. The resulting objective values are: $C_{\max} = 14$, $\sum_j T_j = 4$, $T_{\max} = 2$, and $\sum_j U_j = 2$. In contrast, Figure 3c illustrates the result of Dynamic Scheduling. Here, when Job 1 completes stage 1, it immediately proceeds to stage 2 and is assigned to machine $M_{2,1}$. Later, when $M_{2,1}$ becomes available, both Job 2 and Job 3 are waiting. Since Job 2 has higher priority according to s , it is selected for processing before Job 3. This dynamic adjustment achieves the same makespan as List Scheduling, $C_{\max} = 11$, but crucially, it was able to uphold Job 2's priority, eliminating all tardiness: $\sum_j T_j = 0$.

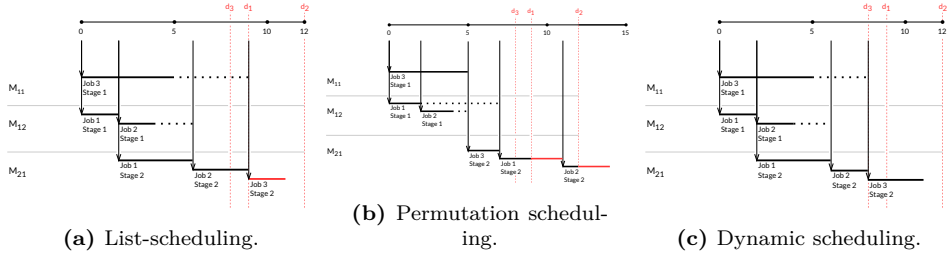


Figure 3. Different decoding methods.

4. Examined algorithms

This paper focuses on three prominent methods: Simulated Annealing (SA), Genetic Algorithms (GA), and Memetic Algorithms (MA).

Simulated Annealing (SA), introduced by Kirkpatrick et al. [13] and Cerny [3], is a probabilistic variant of hill climbing that accepts worsening moves with decreasing probability. Inspired by thermodynamic cooling, it explores a cost landscape via an inhomogeneous Markov chain. The cooling schedule $T(t)$ is a monotonic function impacting convergence quality [16]. For benchmarking, Multiplicative Exponential cooling was used.

Genetic algorithms are search algorithms that mimic natural selection and genetics [11]. The Swap2, Swap3, Adjacent, Reverse segment, Shift segment, Shuffle segment mutations were considered. *Recombination operators* combine parts of two parent solutions to generate one or more offspring. The following operators were considered: Order 1 Crossover (OX1) [4, 9], Order 2 Crossover (OX2) [7, 9], Partially Mapped Crossover (PMX) [8, 9], Cycle Crossover (CX) [18], Edge Recombination Crossover [27].

Selection mechanisms fall into fitness-proportionate (e.g., roulette) and ordinal (e.g., tournament, truncation) categories [7].

Memetic Algorithms (MA) enhance GAs by integrating local search. Inspired by Moscato's model [17], they apply local refinement to globally guided search, improving both quality and convergence speed [25]. Hybrids with nested SA as inner search and MA as outer search have demonstrated superior approximation

performance [1, 6, 28]. Figures 4a and 4b illustrate the core steps of GA and MA, respectively.

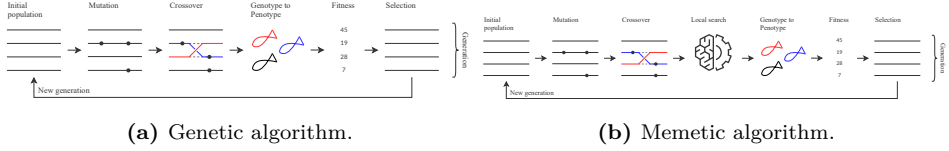


Figure 4. Comparison of memetic and genetic algorithms.

For multi-objective optimization a modified relative distance method was used [14]. The method utilizes two feasible solutions simultaneously and generates a numeric value based on their relative distances. This value is derived from K fitness function value pairs, which are compared and scaled. These scaled values are then multiplied by a weight that signifies an objective's importance. The resulting scaled and multiplied values are then summed. The signedness of the sum signifies dominance or equality.

$$\begin{aligned}
 F: S^2 &\rightarrow \mathbb{R} \\
 F(s_x, s_y) &= \sum_{i=1}^k D(s_x, s_y) \\
 D(s_x, s_y) &= \begin{cases} \frac{f_i(s_y) - f_i(s_x)}{\max(f_i(s_x), f_i(s_y))} & \text{if } \max(f_i(s_x), f_i(s_y)) \neq 0, \\ 0 & \text{otherwise} \end{cases}
 \end{aligned} \tag{4.1}$$

Our modified version uses an *ideal* point $f_i^* = \min\{f_i(s) : s \in P\}$ to derive a distance for every individual in a population $\mathcal{P}(S)$ and sorts them accordingly – the closer an individual is to the ideal, the better. The D function is also made to use the total of absolute values of each objective function value. Therefore, our $F: S \times \mathcal{P}(S) \rightarrow \mathbb{R}$ and $D: S \times \mathcal{P}(S) \rightarrow \mathbb{R}$ can be applied to an entire population. Equation (4.2) formulates our modified approach as opposed to the original method detailed in Equation (4.1).

$$\begin{aligned}
 F(s_x, P) &= \sum_{i=1}^k D(s_x, P) \\
 D(s_x, P) &= \begin{cases} \frac{f_i^* - f_i(s_x)}{|f_i^*| + |f_i(s_x)|} & \text{if } |f_i^*| + |f_i(s_x)| \neq 0, \\ 0 & \text{otherwise} \end{cases}
 \end{aligned} \tag{4.2}$$

5. Benchmark sets

The basis of all benchmarks was the dataset published by Taillard [24]. To generate machine configurations (work centres), constants from the set 1, 2, 4 were selected,

resulting in two scenarios: a *standard case*, where all stages have the same number of machines, and a *bottleneck case*, where the last stage contains only a single machine, simulating a bottleneck in the production flow. Notably, when all stages consist of a single machine, the problem reduces to the classical flowshop variant.

Due dates were generated with a looseness factor $l = 1.3$, using the upper bound UB of each problem as a reference. The due date bounds were computed as $d_{lb} = 0.75 \cdot UB \cdot l$ and $d_{ub} = 1.25 \cdot UB \cdot l$. Each due date d_j was then sampled uniformly as an integer from the interval $[d_{lb}, d_{ub}]$. Setup times $s_{i,j,k}$ were generated randomly as integers in the range $[0, 20]$. For simplicity, total eligibility was assumed – i.e., all jobs are eligible for all machines at every stage.

All random generation was performed using C++’s `std::mt19937` pseudo-random number generator, initialized with the original seed used by Taillard.

6. Results

All mutation, crossover combinations were run on all problems in the 20 job, 5 machine Taillard set (`1tai20_5`). All algorithms were run 10 times – totaling 8100 runs – with the following parameters:

Table 2. Benchmarking parameters.

Parameter	Value
GA generations	1000
MA generations	100
Chromosome count (GA & MA)	20
SA iteration count (MA)	100
SA initial temperature	3000
SA α	0.1

Distance values were calculated similarly to Equation (4.2), using the entire result set from all algorithms on a specific problem, stage, bottleneck configuration. Figure 5 clearly shows how memetic algorithms edge out genetic algorithms on standard benchmarks and Inversion mutation was the operator that benefitted the most from the introduction of a local search algorithm.

A percentile difference ω from the best known solution is also used. The measure for comparison is the upper bound, the best solution known so far. From the obtained final makespan C_{BS} and the upper bound C_{UB} , a difference is calculated $\omega = \frac{C_{BS} - C_{UB}}{C_{UB}} \cdot 100$.

Table 3 presents the best configurations for Genetic Algorithms (GA) and Memetic Algorithms (MA) across varying numbers of machines per stage and bottleneck settings. Configurations were selected based on the lowest total distance from ideal objective values. For each condition, the GA and MA configurations are compared side by side to highlight their relative performance.

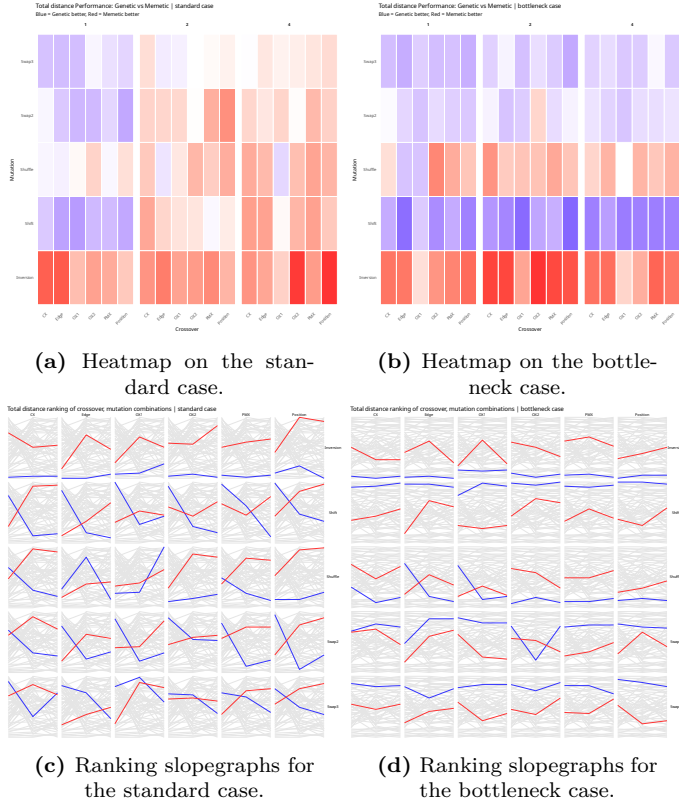


Figure 5. Relative distance comparison per mutation, recombination for each machine count on standard and bottleneck cases.

In the case of one machine per stage without bottlenecking, the Memetic Algorithm using the Inversion/CX combination outperforms its genetic counterpart in all metrics. It achieves a lower makespan (C_{max}), reduced total tardiness ($\sum T$), fewer late jobs ($\sum U$), and a lower ω value, although the genetic configuration with Shift/OX1 remains competitive with a marginally better total distance. Under the same machine configuration but with a bottleneck present, the Genetic Algorithm demonstrates superior performance across all evaluated objectives. The configuration using Shift/Position achieves the best results, with minimal C_{max} , tardiness, and late jobs, as well as the lowest ω and total distance, indicating GA's adaptability in constrained environments at this scale.

With two machines per stage and no bottleneck, both algorithms perform near optimally. The Genetic Algorithm with the Swap3/OX1 configuration achieves perfect scheduling, indicated by zero tardiness and no late jobs. The Memetic Algorithm also reaches optimality with a nearly identical performance, showing both methods to be equally effective under these conditions.

Table 3. Best configurations by total distance per stage.

Alg.	M ¹	B ²	C_{max}	$\sum T$	$\sum U$	ω	F	Config ³
GA	1	F	1712.82	302.31	3.06	40.70	0.01	Shift/OX1
MA	1	F	1636.28	245.25	2.43	34.23	0.02	Inversion/CX
GA	1	T	1693.31	254.16	2.43	38.76	0.01	Shift/Position
MA	1	T	1723.82	282.51	3.06	41.50	0.02	Shuffle/CX
GA	2	F	981.94	0.00	0.00	-19.39	0.01	Swap3/OX1
MA	2	F	985.26	0.00	0.00	-19.18	0.01	Inversion/Position
GA	2	T	1379.80	42.93	0.63	13.05	0.18	Shift/Position
MA	2	T	1389.63	69.48	0.72	13.72	0.19	Shift/OX2
GA	4	F	609.87	0.00	0.00	-49.98	0.01	Shuffle/OX1
MA	4	F	607.59	0.00	0.00	-50.19	0.01	Shuffle/Position
GA	4	T	1356.70	56.88	0.36	10.95	0.39	Shift/OX2
MA	4	T	1377.07	30.60	0.54	12.79	0.40	Swap2/Edge

When a bottleneck is introduced at the same stage count, the Genetic Algorithm once again proves more robust. Its Shift/Position configuration yields better results in every objective, outperforming the memetic counterpart and reinforcing GA's advantage under more constrained and complex processing.

At four machines per stage with no bottleneck, the Memetic Algorithm takes the lead. The configuration with Shuffle/Position outperforms GA's best setup by achieving slightly better values for C_{max} , ω , and total distance, suggesting that memetic search strategies scale more effectively with increased stage parallelism.

Finally, in the most complex case – four machines per stage with a bottleneck – both algorithms show strengths in different aspects. The Genetic Algorithm configuration yields a better makespan and fewer late jobs, while the Memetic Algorithm significantly reduces total tardiness and produces a more favorable ω value. Although the Genetic Algorithm has a marginally better total distance, the overall results indicate a trade-off between the two strategies depending on the specific scheduling objective.

These findings suggest that Genetic Algorithms tend to perform better in low-stage or bottlenecked environments, while Memetic Algorithms excel as system complexity increases and resources are less constrained. The mutation and crossover pairings also play a critical role, with combinations like Shift/Position, Shuffle/OX1, and Inversion/CX consistently appearing among the top-performing configurations.

Selection of a parameter set for Memetic Algorithms must also consider the runtime overhead introduced by the additional local search operation. This creates a balancing act in CPU time-management between local and global search.

¹Machines per stage

²Bottleneck configuration

³Mutation/Crossover

7. Conclusion

This study presented a comparative evaluation of Genetic and Memetic Algorithms for solving flexible flowshop scheduling problems under varying machine stage counts and bottleneck conditions. The results indicate that Genetic Algorithms tend to perform better in constrained environments, especially when bottlenecks are present or stage counts are low, while Memetic Algorithms show superior scalability and robustness as the degree of parallelism increases. The evaluation across multiple objective functions – makespan, total tardiness, number of late jobs, and an aggregate ω metric – highlighted the importance of pairing crossover and mutation operators effectively with the problem structure.

8. Future research

These findings are compelling and provides ample room for future research. Future work will explore several extensions of this study, including integrating both fine-grained and coarse-grained parallel algorithms to reduce runtime and improve scalability. Adding realistic constraints such as setup times, machine eligibility, and maintenance. Combining metaheuristics with Constraint Programming could offer better feasibility guarantees. Using neural networks to guide job and machine selection within metaheuristics, enabling adaptive scheduling policies. Extending with algorithms to explore trade-offs between makespan, tardiness, and other objectives.

References

- [1] A. AGÁRDI, K. NEHÉZ, O. HORNYÁK, L. T. KÓCZY: *A Hybrid Discrete Bacterial Memetic Algorithm with Simulated Annealing for Optimization of the Flow Shop Scheduling Problem*, Symmetry 13.7 (2021), p. 1131, DOI: [10.3390/sym13071131](https://doi.org/10.3390/sym13071131).
- [2] S. AGHAKHANI, M. S. RAJABI: *A new hybrid multi-objective scheduling model for hierarchical hub and flexible flow shop problems*, AppliedMath 2.4 (2022), pp. 721–737, DOI: [10.3390/appliedmath2040043](https://doi.org/10.3390/appliedmath2040043).
- [3] V. ČERNÝ: *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm*, Journal of optimization theory and applications 45 (1985), pp. 41–51, DOI: [10.1007/BF00940812](https://doi.org/10.1007/BF00940812).
- [4] L. DAVIS ET AL.: *Applying adaptive algorithms to epistatic domains*. In: IJCAI, vol. 85, Citeseer, 1985, pp. 162–164, ISBN: 0934613028.
- [5] K. DEB, M. EHRGOTT: *On Generalized Dominance Structures for Multi-Objective Optimization*, Mathematical and Computational Applications 28.5 (2023), ISSN: 2297-8747, DOI: [10.3390/mca28050100](https://doi.org/10.3390/mca28050100), URL: <https://www.mdpi.com/2297-8747/28/5/100>.
- [6] L. FAZEKAS, B. TÜÜ-SZABÓ, L. T. KÓCZY, O. HORNYÁK, K. NEHÉZ: *A Hybrid Discrete Memetic Algorithm for Solving Flow-Shop Scheduling Problems*, Algorithms 16.9 (2023), ISSN: 1999-4893, DOI: [10.3390/a16090406](https://doi.org/10.3390/a16090406), URL: <https://www.mdpi.com/1999-4893/16/9/406>.
- [7] D. E. GOLDBERG, B. KORB, K. DEB: *Messy genetic algorithms: Motivation, analysis, and first results*, Complex systems 3.5 (1989), pp. 493–530, ISSN: 0891-2513.

- [8] D. E. GOLDBERG, R. LINGLE JR: *Alleles, Loci, and the Traveling Salesman Problem*, in: Proceedings of the 1st International Conference on Genetic Algorithms, 1985, pp. 154–159, DOI: [10.4324/9781315799674](https://doi.org/10.4324/9781315799674).
- [9] Y. GUAN, Y. CHEN, Z. GAN, Z. ZOU, W. DING, H. ZHANG, Y. LIU, C. OUYANG: *Hybrid flow-shop scheduling in collaborative manufacturing with a multi-crossover-operator genetic algorithm*, Journal of Industrial Information Integration 36 (2023), p. 100514, ISSN: 2452-414X, DOI: [10.1016/j.jii.2023.100514](https://doi.org/10.1016/j.jii.2023.100514), URL: <https://www.sciencedirect.com/science/article/pii/S2452414X23000870>.
- [10] J. N. GUPTA, S. K. GUPTA: *Single facility scheduling with nonlinear processing times*, Computers & Industrial Engineering 14.4 (1988), pp. 387–393, ISSN: 0360-8352, DOI: [10.1016/0360-8352\(88\)90041-1](https://doi.org/10.1016/0360-8352(88)90041-1), URL: <https://www.sciencedirect.com/science/article/pii/0360835288900411>.
- [11] J. H. HOLLAND: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992, ISBN: 9780262275552, DOI: [10.7551/mitpress/1090.001.0001](https://doi.org/10.7551/mitpress/1090.001.0001).
- [12] S. M. JOHNSON: *Optimal two-and three-stage production schedules with setup times included*, Naval research logistics quarterly 1.1 (1954), pp. 61–68, DOI: [10.1002/nav.3800010110](https://doi.org/10.1002/nav.3800010110).
- [13] S. KIRKPATRICK, C. D. GELATT JR, M. P. VECCHI: *Optimization by simulated annealing*, science 220.4598 (1983), pp. 671–680, DOI: [10.1126/science.220.4598.671](https://doi.org/10.1126/science.220.4598.671).
- [14] G. KULCSÁR, F. ERDÉLYI: *A new approach to solve multi-objective scheduling and rescheduling tasks*, International Journal of Computational Intelligence Research 3.4 (2007), pp. 343–351.
- [15] K. MIHÁLY, G. KULCSÁR: *A New Many-Objective Hybrid Method to Solve Scheduling Problems*, International Journal of Industrial Engineering and Management 14.4 (2023), pp. 326–335, DOI: [10.24867/IJIEEM-2023-4-342](https://doi.org/10.24867/IJIEEM-2023-4-342).
- [16] J. MILICZKI, L. FAZEKAS: *Comparison of Cooling Strategies in Simulated Annealing Algorithms for Flow-shop Scheduling*, Production Systems and Information Engineering 10.3 (2022), pp. 129–136, DOI: [10.32968/psaie.2022.3.10](https://doi.org/10.32968/psaie.2022.3.10).
- [17] P. MOSCATO: *On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms*, Technical Report, Caltech Concurrent Computation Program Report 826, (1989).
- [18] I. M. OLIVER, D. J. SMITH, J. R. C. HOLLAND: *A study of permutation crossover operators on the traveling salesman problem*, in: Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application, Cambridge, Massachusetts, USA: L. Erlbaum Associates Inc., 1987, pp. 224–230, ISBN: 0805801588.
- [19] A. OPRIS, D.-C. DANG, F. NEUMANN, D. SUDHOLT: *Runtime Analyses of NSGA-III on Many-Objective Problems*, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '24, Melbourne, VIC, Australia: Association for Computing Machinery, 2024, pp. 1596–1604, ISBN: 9798400704949, DOI: [10.1145/3638529.3654218](https://doi.org/10.1145/3638529.3654218).
- [20] M. L. PINEDO: *Scheduling, Theory, Algorithms, and Systems*, 6th ed., Springer Cham, 2022, ISBN: 9783031059216, DOI: [10.1007/978-3-031-05921-6](https://doi.org/10.1007/978-3-031-05921-6).
- [21] I. RIBAS, R. LEISTEN, J. M. FRAMÍÑAN: *Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective*, Computers & Operations Research 37.8 (2010), Operations Research and Data Mining in Biological Systems, pp. 1439–1454, ISSN: 0305-0548, DOI: [10.1016/j.cor.2009.11.001](https://doi.org/10.1016/j.cor.2009.11.001), URL: <https://www.sciencedirect.com/science/article/pii/S0305054809002883>.
- [22] R. RUIZ, T. STÜTZLE: *A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem*, European Journal of Operational Research 177.3 (2007), pp. 2033–2049, ISSN: 0377-2217, DOI: [10.1016/j.ejor.2005.12.009](https://doi.org/10.1016/j.ejor.2005.12.009), URL: <https://www.sciencedirect.com/science/article/pii/S0377221705008507>.

- [23] R. RUIZ, J. A. VÁZQUEZ-RODRÍGUEZ: *The hybrid flow shop scheduling problem*, European Journal of Operational Research 205.1 (2010), pp. 1–18, ISSN: 0377-2217, DOI: [10.1016/j.ejor.2009.09.024](https://doi.org/10.1016/j.ejor.2009.09.024), URL: <https://www.sciencedirect.com/science/article/pii/S0377221709006390>.
- [24] E. TAILLARD: *Benchmarks for basic scheduling problems*, European Journal of Operational Research 64.2 (1993), Project Management and Scheduling, pp. 278–285, ISSN: 0377-2217, DOI: [10.1016/0377-2217\(93\)90182-M](https://doi.org/10.1016/0377-2217(93)90182-M), URL: <https://www.sciencedirect.com/science/article/pii/037722179390182M>.
- [25] B. TÜÜ-SZABÓ, P. FÖLDESI, L. T. KÓCZY: *An efficient evolutionary metaheuristic for the traveling repairman (minimum latency) problem*, International Journal of Computational Intelligence Systems 13.1 (2020), pp. 781–793, DOI: [10.2991/ijcis.d.200529.001](https://doi.org/10.2991/ijcis.d.200529.001).
- [26] T. URLINGS, R. RUIZ, F. S. SERIFOGLU: *Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems*, International Journal of Metaheuristics 1.1 (2010), pp. 30–54, DOI: [10.1504/IJMHeur.2010.033122](https://doi.org/10.1504/IJMHeur.2010.033122).
- [27] L. D. WHITLEY, T. STARKWEATHER, D. FUQUAY: *Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator*, in: Proceedings of the 3rd International Conference on Genetic Algorithms, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 133–140, ISBN: 1558600663.
- [28] A. J. WILSON, D. PALLAVI, M. RAMACHANDRAN, S. CHINNASAMY, S. SOWMIYA: *A review on memetic algorithms and its developments*, Electrical and Automation Engineering 1.1 (2022), pp. 7–12, DOI: [10.46632/eae/1/1/2](https://doi.org/10.46632/eae/1/1/2).
- [29] C. YU, Q. SEMERARO, A. MATTA: *A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility*, Computers & Operations Research 100 (2018), pp. 211–229, ISSN: 0305-0548, DOI: [10.1016/j.cor.2018.07.025](https://doi.org/10.1016/j.cor.2018.07.025), URL: <https://www.sciencedirect.com/science/article/pii/S030505481830217X>.

Automated detection of toxic comments in Hungarian

Péter Hatvani^{ab}, Zijian Győző Yang^a

^aELTE Research Centre for Linguistics
yang.zijian.gyozo@nytud.elte.hu

^bPázmány Péter Catholic University
Doctoral School of Linguistics
hatvani.peter@hallgato.ppke.hu

Abstract. Moderating toxic online comments in Hungarian remains a challenging NLP task. We introduce the first openly available Hungarian corpus for toxic comment classification, though limited in size ($n = 655$), sourced from social media and political news forums. We fine-tuned three BERT-based classifiers (huBERT, multilingual BERT, and huBERT-SetFit) and applied data augmentation techniques to expand the training dataset. The best-performing model, huBERT-SetFit, achieved an F1 score of 93.7%. Our results demonstrate the effectiveness of transformer-based models for toxicity detection in low-resource, linguistically complex settings.

Keywords: toxicity, online hate, nlp, classification, logistic regression

AMS Subject Classification: 68T07, 68T30, 68T50, 91F20

1. Introduction

In the digital era, online platforms have become central to social interaction, yet they are frequently plagued by toxic discourse that undermines constructive engagement. Toxic comments can include hate speech, threats, personal insults, and discriminatory remarks, posing challenges to both platform moderation and user safety. Although large-scale datasets and models exist for high-resource languages such as English, many low-resource languages such as Hungarian remain underrepresented in this domain.

This paper addresses the critical need for effective toxicity detection in Hungarian by introducing a novel, publicly available corpus containing both toxic and

neutral Hungarian comments. Additionally, we evaluate the performance of three fine-tuned transformer-based classifiers – HuBERT, multilingual BERT (mBERT), and huBERT-SetFit – on this dataset.

Drawing inspiration from previous efforts such as detoxify [3] and the Jigsaw Multilingual Toxicity Classification competition [5], this research aims to bring similar capabilities to the Hungarian language. Our experiments show that sentence embedding-based methods like SetFit can outperform traditional fine-tuning in low-data regimes, providing a viable solution for toxicity detection in under-resourced languages.

2. Related works

The detection of toxic comments has gained increasing attention with the growth of user-generated content. One of the most influential initiatives in this space was led by the Jigsaw/Conversation AI team, which introduced large-scale English-language datasets for toxicity classification on platforms such as Wikipedia talk pages [5]. These datasets provided multilabel annotations for categories such as: Toxic, Severe toxic, Obscene, Threat, Insult, Identity hate.

Several benchmark models such as Toxic-BERT [3], based on pretrained transformer architectures, have since been developed and are widely used in high-resource English and multilingual contexts. However, these models often perform inadequately in underrepresented languages due to data scarcity and linguistic differences.

Multilingual BERT [2] and XLM-R [1] offer some generalization to low-resource languages, but studies show that language-specific models like huBERT [6] can outperform them in Hungarian-specific tasks.

SetFit [8] introduced a new paradigm by combining sentence embeddings with lightweight classification heads, enabling few-shot learning with minimal labeled data. Its efficiency and performance in low-resource settings make it particularly suited for tasks such as toxicity detection in Hungarian.

Despite these advances, there remains a lack of open-domain Hungarian datasets and benchmark models for toxic comment classification. Our work contributes to filling this gap by releasing a small but diverse Hungarian dataset and evaluating three transformer-based models on it, including a SetFit variant that requires no data augmentation.

3. Method

To develop an automatic classifier for toxic comment detection, a manually annotated dataset was first created. The training corpus comprises comments from two distinct sources: (i) offensive social media comments from *Reddit* and *napiszar.com*, and (ii) politically charged discussions from Hungarian news sites *mandiner.hu* and

kuruc.info. The final dataset comprises 655 annotated comments, labeled according to the jigsaw competition categories.

3.1. Annotation categories and examples

We have adopted the multi-label approach from Jigsaw with the 5 scale Likert scale to the same categories as mentioned in the Related Works section. Three annotators made judgements with a “slight-agreement” according to Randolph’s $\kappa_{\text{free}} = 0.525$ [7]. The annotators were not in the same age group neither were they all of the same gender. Given the subjective nature of toxicity perception, we expected substantial variation in judgments across items when using annotators from diverse backgrounds. Although judgements were not unanimous, no item exhibited extreme disagreement (with one annotator giving the highest rating and another giving the lowest), so no items were removed from the initial collection.

While the moderate inter-annotator agreement reflects the inherent subjectivity in toxicity perception, this represents a limitation of our dataset that may affect model reliability. Future work should explore consensus-building techniques or expert adjudication to improve annotation consistency.

3.2. Analysis of examples from the corpus

Toxicity can show many forms. In this section, we introduce a few examples from the corpus and analyse the toxic content in each.

- (1) *Ezt az arcot láttam már valahol, mintha egy híg*
 this.ACC the face-ACC see-1SG.PAST already somewhere as.if a watery
agyú propaganda-troll lenne.
 brain-POSS propaganda-troll be.COND.3SG

‘I’ve seen this face somewhere before, like some dumb-brained propaganda troll.’

Toxicity Example 1 conveys toxicity through indirect derogation. The speaker insinuates that the person resembles a “dumb-brained propaganda troll,” a phrase that implies intellectual inferiority and political manipulation. The use of “mintha” (as if) introduces the insult in a covert, sarcastic way, enhancing its rhetorical impact while allowing for plausible deniability.

- (2) *Kárpótol addig hatalmas masztika, míg nem jut pina.*
 compensate until huge jerk.off while not get pussy

‘Just jerk off until you can get some pussy.’

Obscenity This utterance in Example 2 is obscene and explicitly sexual, using vulgar slang for masturbation and female genitalia. It expresses objectification and dehumanization, especially through the reduction of sexual partners to means of “compensation.” The aggressive tone and crude lexicon contribute to its toxic nature, targeting both sexual frustration and women.

- (3) *Írjon naked a boncmester, jegyzőkönyvet.*
 write-IMP.3SG you.DAT the autopsy-master report-ACC
 ‘Let the coroner write your report.’

Threat This sentence in Example 3 implies death in a sarcastic manner by suggesting that a coroner should write the recipient’s report. While indirect, it is a menacing wish for harm cloaked in formal-sounding language. The choice of a professional figure related to death underscores the implied violent intent without making an overt threat.

- (4) *Attól hogy blokkolod, még mindig igazán van, és te*
 from.that that block-DERIV-2SG still always true-POSS.3SG is and you
továbbra is egy idióta vagy.
 still a idiot are

‘Blocking them doesn’t change the fact they’re right, and you’re still an idiot.’

Insult This comment in Example 4 employs direct personal insult by calling the recipient an “idiot.” It also undermines their actions (blocking someone) as ineffective, intensifying the disparagement. The phrasing suggests moral superiority while dismissing the recipient’s perspective, reinforcing the toxic tone through condescension and name-calling.

- (5) *Bárcsak minden buzit kirúgnának!!!!*
 if.only all faggot-ACC out-kick-COND.3SG-PL
 ‘If only all the faggots got fired!!!!’

Identity Hate This example in Example 5 is a clear instance of hate speech targeting a marginalized identity group. The slur “buzit” (faggot) is used with a wish for collective punishment (“get fired”), which constitutes discriminatory rhetoric. The multiple exclamation marks amplify the emotional aggression, highlighting the severity of the hate expressed.

- (6) *Úgy-úgy, kis hülye! Csússzál térden előttük. A kurva*
 so-so little idiot crawl-IMP.2SG knee-SUP before-3PL the whore
anyádat te agyhalott pondró.
 mother-POSS.2SG-ACC you brain-dead maggot

‘Yeah right, you little idiot! Crawl on your knees in front of them. Fuck your mother, you brain-dead maggot.’

Severe Toxicity This utterance in Example 6 combines multiple forms of toxicity: insult (“idiot,” “brain-dead maggot”), obscenity (“fuck your mother”), and verbal domination (“crawl on your knees”). It escalates through commands and extreme invective. The layered abuse represents severe toxicity, intended to intimidate, degrade, and humiliate the addressee completely.

3.3. Training data preparation

As previously discussed, our manually collected toxic corpus helps capture the nuances of Hungarian cultural context in toxic language. However, given the small size of our original corpus, we supplemented it with the Hungarian Twitter corpus¹ as a source of neutral examples.

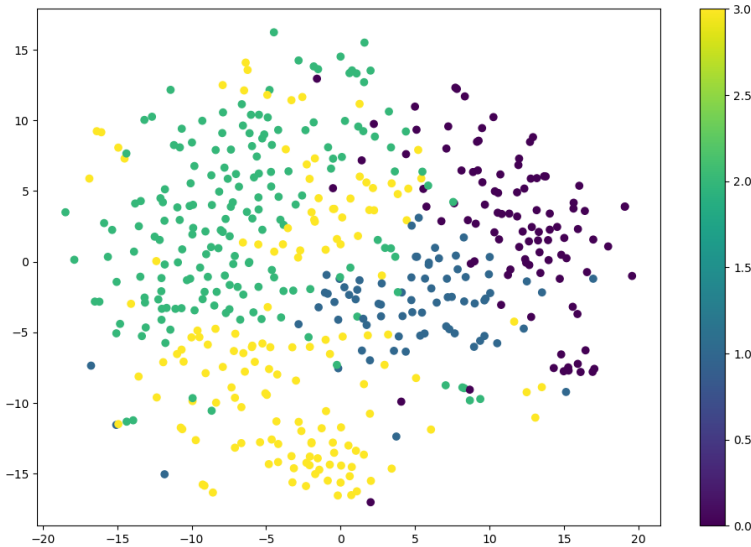


Figure 1. K-means clustering of the comments.

¹<https://opendata.hu/dataset/hungarian-twitter-sentiment-corpus> – property of Precognox Ltd.

Clustering of the toxic corpus To better understand the structure of our toxic comment corpus, we visualized huBERT-generated sentence embeddings using t-SNE dimensionality reduction (Figure 1). We then applied K-means clustering with $k=4$, motivated by the four distinct data sources: comments from *kuruc.info*, *mandiner.hu*, the *napiszar.com* shock site, and Reddit posts. As shown in Figure 1, the resulting clusters reveal meaningful groupings in the embedded space. Although some overlap exists (common in language data), the clusters are generally well-formed and reflect source-specific patterns in toxic language use. In particular, the Reddit and *napiszar.com* content forms broader, more dispersed clusters, likely due to their conversational and informal nature, while politically charged comments from news sites appear more tightly grouped.

Hungarian Twitter Sentiment Corpus The *Hungarian Twitter Sentiment Corpus* (HTS) is a publicly available collection of approximately 4,000 Hungarian-language tweets annotated for sentiment polarity. The corpus was created by PrecognoX and made accessible through opendata.hu. It includes five sentiment labels on a Likert-like scale, ranging from 1 (very negative) to 5 (very positive), which form the HTS5 variant. A binary version (HTS2) was also derived by grouping positive and negative classes and excluding neutral tweets. In our work, we used HTS5 labels and treated tweets rated 3–5 as *neutral* training data, following a pragmatic interpretation where mid-scale ratings reflect low-intensity sentiment or ambiguous tone. This decision enabled a clearer separation between offensive and non-offensive language in our toxicity classification task. The corpus provides a valuable resource for sentiment modeling in Hungarian, and our adaptation aligns with common practices in low-resource language scenarios where neutral and ambiguous categories are often merged to improve class balance and model performance.

Models trained and evaluated There are a plethora of models available for moderation. Most of them are products of companies that are available for a fee per request or token. To establish a baseline, we have evaluated the models finetuned by us with OpenAI’s Moderation API. The API in question is *omni-moderation-latest* that was available on 2025.07.02. We have evaluated four models collected in the list 3.3 from which we fine-tuned three of the models. The OpenAI endpoint was the baseline for the evaluation for it is widely used moderation tool for AI models currently. The *toxic-hubert* model was fine-tuned from the HuBERT model [6], same method as for the multilingual BERT. The *hubert-embedding-setfit-toxic* model was fine-tuned from an embedding model [4] with the SetFit toolset.

- Openai Moderation endpoint (*omni-moderation-latest*)
- RabidUmarell/*toxic-hubert*
- RabidUmarell/*toxic-mbert*
- RabidUmarell/*hubert-embedding-setfit-toxic*

Models were evaluated using standard classification metrics including precision, recall, F1-score, and accuracy. Statistical significance was assessed using McNemar’s test with $\alpha = 0.05$.

Training Data Augmentation Before training, we have applied two data augmentation techniques: Typographical error simulation and masked token replacement to increase the training set size, resulting in 10,185 toxic and 17,266 neutral instances.

Training Each model was trained with standard BERT hyperparameters: learning rate $\eta = 2 \cdot 10^{-5}$, batch size of 8, weight decay of 0.01, and mixed precision training using `fp16`. The SetFit model, on the contrary, did not require data augmentation; it used the huBERT sentence transformer [4] to generate embeddings and used a lightweight logistic regression head for classification. This approach aligns with SetFit’s promise of achieving high performance with minimal computational overhead, particularly in low-resource scenarios. A detailed summary of training performance can be seen in Table 1. The HuBERT model was only trained for one epoch because this model reached equilibrium quickly and additional epochs only degraded the performance.

Table 1. Training and validation losses and F1 scores of different models.

Model	Epochs	Training Loss	F1 Score
HuBERT	1	0.317000	0.873582
mBERT	3	0.593200	0.790007
huBERT-embedding-setfit-toxic	3	0.2175	0.93725

4. Results

We evaluated the trained models on a small test dataset (available²)

Toxicity Classification Accuracy Comparison We evaluated four models (SetFit Toxic-HuBERT, Toxic-HuBERT, Toxic-mBERT, and OpenAI Moderations API) across eight manually annotated toxicity categories plus a neutral control condition. The SetFit Toxic-HuBERT model achieved consistently strong performance, reaching perfect or near-perfect accuracy in *Hate Speech*, *Threat*, *Obscenity* / *Profanity*, and *Harassment* / *Bullying* (100% in all these categories except for a minor drop in *Toxic Generalization*, 75%).

Toxic-mBERT closely followed, also achieving 100% accuracy in four toxicity categories and demonstrating solid generalization. Meanwhile, Toxic-HuBERT showed

²<https://huggingface.co/datasets/RabidUmarell/hu-toxic-test-set>

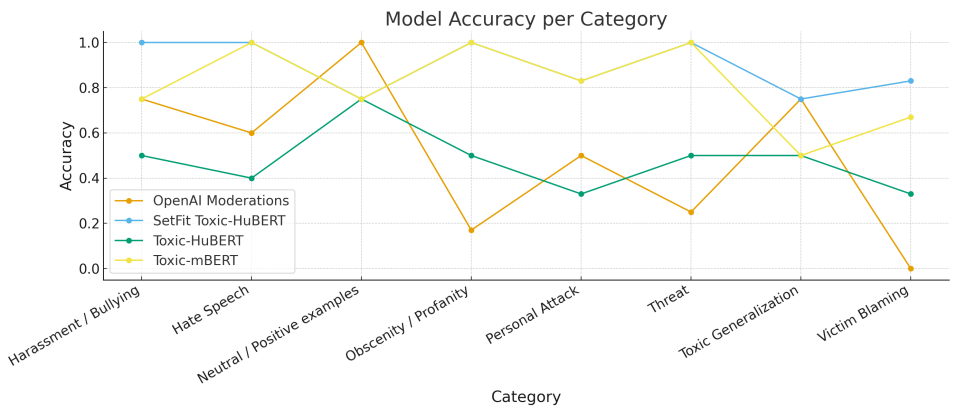


Figure 2. Model accuracy on the test set.

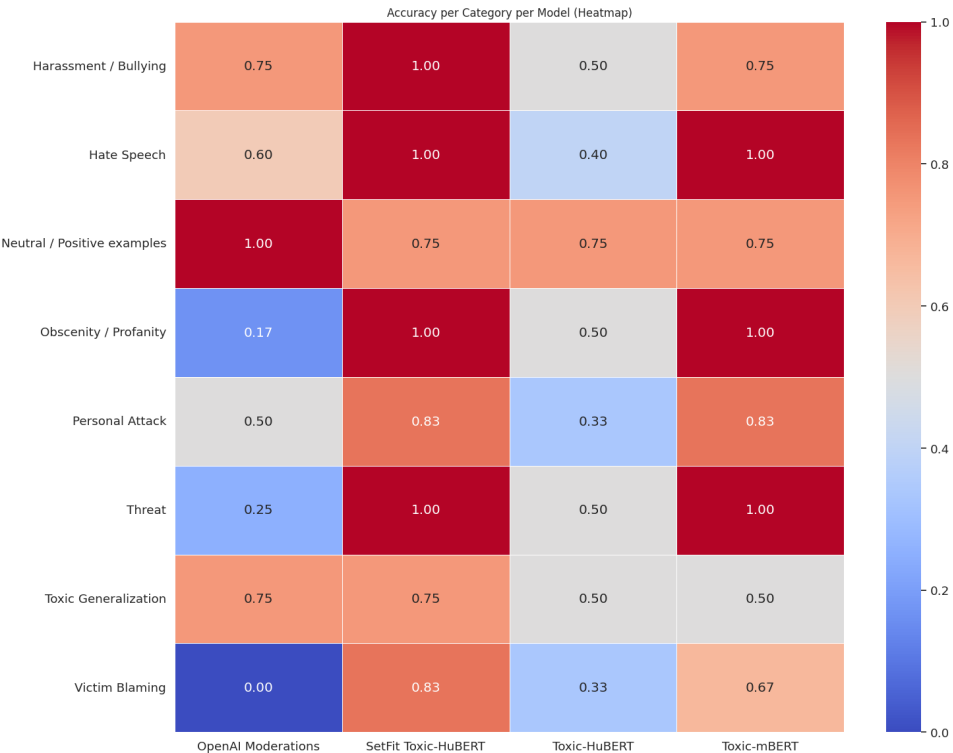


Figure 3. Heatmap of the models' performance on the test set.

weaker overall performance, with accuracy between 33%–75% across all categories, suggesting lower confidence or overfitting. In contrast, the **OpenAI Moderations**

API displayed a more varied performance profile: high accuracy in broader categories like *Personal Attack* (50%), *Hate Speech* (60%), and *Harassment / Bullying* (75%), but weaker results in more nuanced cases such as *Victim Blaming* (0%) and *Threat* (25%).

Neutral examples were handled best by **OpenAI Moderations** and **Toxic-HuBERT** (100%), indicating reliable non-toxic classification. Overall, **SetFit Toxic-HuBERT** and **Toxic-mBERT** emerged as the most balanced models, with robust performance in detecting multiple forms of toxicity across languages and expressions.

Statistical significance was assessed using McNemar’s test for paired comparisons. The huBERT-SetFit model significantly outperformed both huBERT ($p < 0.01$) and mBERT ($p < 0.05$), while the difference between huBERT and mBERT was not statistically significant ($p = 0.12$).

5. Conclusion

This paper introduced a novel, manually annotated Hungarian dataset for toxic comment classification and presented a comparative evaluation of three transformer-based models – huBERT, mBERT, and SetFit – on this task. Our experiments demonstrate that sentence embedding-based approaches, particularly SetFit combined with huBERT, offer strong and reliable performance across a wide range of toxicity categories, even in low-data conditions. Notably, SetFit achieved high accuracy without requiring data augmentation, confirming its utility in low-resource scenarios.

Our results indicate that while general-purpose multilingual models like mBERT provide reasonable baseline performance, Hungarian-specific models better handle the morphological complexity and cultural nuances of Hungarian toxic language. The huBERT-based SetFit model consistently outperformed traditional fine-tuned counterparts, especially in categories such as obscenity, threat, and identity hate, where subtle linguistic cues play a key role.

Several limitations should be acknowledged. The relatively small dataset size (655 comments) may limit generalizability, and the moderate inter-annotator agreement ($\kappa_{free} = 0.525$) suggests inherent challenges in toxicity annotation. Additionally, the cultural and platform-specific nature of our data sources may not fully represent the diversity of Hungarian toxic language across all digital contexts.

By releasing both the annotated corpus and the model evaluation results, this work contributes a much-needed resource for Hungarian NLP and opens the door to further research on toxicity detection in underrepresented languages. Beyond the immediate task, the dataset and findings also provide a foundation for developing safer and more context-aware content moderation tools in Hungarian digital spaces. In doing so, our work supports broader efforts toward building inclusive, multilingual language technologies that reflect the full diversity of online communication.

References

- [1] A. CONNEAU, K. KHANDLWAL, N. GOYAL, V. CHAUDHARY, G. WENZKE, F. GUZMÁN, E. GRAVE, M. OTT, L. ZETTLEMOYER, V. STOYANOV: *Unsupervised Cross-lingual Representation Learning at Scale*, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ed. by D. JURAFSKY, J. CHAI, N. SCHLUTER, J. TETREAULT, Online: Association for Computational Linguistics, July 2020, pp. 8440–8451, DOI: [10.18653/v1/2020.acl-main.747](https://doi.org/10.18653/v1/2020.acl-main.747), URL: <https://aclanthology.org/2020.acl-main.747/>.
- [2] J. DEVLIN, M.-W. CHANG, K. LEE, K. TOUTANOVA: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), ed. by J. BURSTEIN, C. DORAN, T. SOLORIO, Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186, DOI: [10.18653/v1/N19-1423](https://doi.org/10.18653/v1/N19-1423), URL: <https://aclanthology.org/N19-1423>.
- [3] L. HANU, UNITARY TEAM: *Detoxify*, Github. <https://github.com/unitaryai/detoxify>, 2020.
- [4] P. HATVANI, Z. G. YANG: *Training Embedding Models for Hungarian*, in: Proceedings of the 2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS), Debrecen: University of Debrecen, 2024, pp. 75–80, ISBN: 9798350387889.
- [5] I. KIVLICHAN, J. SORENSEN, J. ELLIOTT, L. VASSERMAN, M. GÖRNER, P. CULLITON: *Jigsaw Multilingual Toxic Comment Classification*, 2020, URL: <https://kaggle.com/competitions/jigsaw-multilingual-toxic-comment-classification>.
- [6] D. M. NEMESKEY: *Introducing huBERT*, in: XVII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2021), Szeged, 2021, pp. 3–14.
- [7] J. J. RANDOLPH: *Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss' Fixed-Marginal Multirater Kappa*, in: Joensuu Learning and Instruction Symposium, vol. 2005, 2005, URL: <https://eric.ed.gov/?id=ED490661>.
- [8] L. TUNSTALL, N. REIMERS, U. E. S. JO, L. BATES, D. KORAT, M. WASSERBLAT, O. PEREG: *Efficient Few-Shot Learning Without Prompts*, 2022, DOI: [10.48550/ARXIV.2209.11055](https://doi.org/10.48550/ARXIV.2209.11055), URL: <https://arxiv.org/abs/2209.11055>.

Solving the Team Coordination on Graphs With Risky Edges problem for nonzero self-loop weights*

András Izsó, István Harmati

Budapest University of Technology and Economics
Department of Control Engineering and Information Technology
izso@iit.bme.hu
harmati@iit.bme.hu

Abstract. Multiagent system control is a well-researched area of recent years, since the cooperation of multiple agents opens up the possibility to tackle more complicated problems and create finer scaling systems. Team Coordination on Graphs with Risky Edges has been recently proposed and provides a framework to model such systems. In this problem, multiple agents traverse through a graph. Apart from the ordinary nodes and edges, the graph also contains support nodes, where an agent can choose to support another agent that is moving through a so-called risky edge, associated with the support node. Some solutions have already been proposed; however, all of them assume zero cost of waiting, which is restrictive in many real-world problems. In this paper, we generalize the problem, allowing non-zero cost of waiting, make a solution proposal, and present our comprehensive simulation results.

Keywords: multi-agent, control, cooperation, team, graph

AMS Subject Classification: 93A16 Multi-agent systems, 91A12 Cooperative games

1. Introduction

Multiagent system control is an increasingly popular research area nowadays. The reduction of price and size of hardware opened up the possibility of using multiple,

*This work was supported by the NSF grant DMS 2054735.

The project supported by the Doctoral Excellence Fellowship Programme (DCEP) is funded by the National Research Development and Innovation Fund of the Ministry of Culture and Innovation and the Budapest University of Technology and Economics.

in some sense simpler agents instead of a single, complex one. This leads to better scalability and cost-effectiveness. However, the complexity is not avoided, only moved to a new level, which raises the need for algorithms that are capable of defining the behavior of multiple agents in order to reach the theoretical optimum in practice.

One of the core problems of such systems in robotics is the Multiagent Path Finding [9]. Its main application areas include public transport [1], package delivery [8] and general formation and swarm control [3]. Previously, most of these approaches looked at agents merely as obstacles that might make their own decisions, but mainly they just have to be avoided. The line of research to which we would like to contribute changes this by the introduction of coordination/cooperation, which promotes interaction between agents.

The Team Coordination on Graphs with Risky Edges (TCGRE) problem provides a framework to model multiagent scenarios, where the action of an agent can reduce the cost of the action of another agent. The agents operate on a weighted graph, representing points of physical or state space, and each agent's task is to traverse from their start nodes to their respective goal nodes, inducing the lowest possible cost. Some nodes of the graph, called *support nodes* are associated with certain edges, called *risky edges*. If an agent in one of the support nodes chooses, instead of moving, to support one of the associated risky edges of their current node, the other agent, that is traveling through that edge in the same time step can do so for a reduced cost.

An example graph is given in Figure 1. There are two agents, traveling respectively from S1 to G1 and S2 to G2. Support nodes are C1, C2, and C3; the risky edges are shown in red and the association between them as green dashed arrows. The cooperation enables the agents to diverge from their individual shortest paths (marked as orange and yellow) to achieve lower cost through cooperation (blue and green).

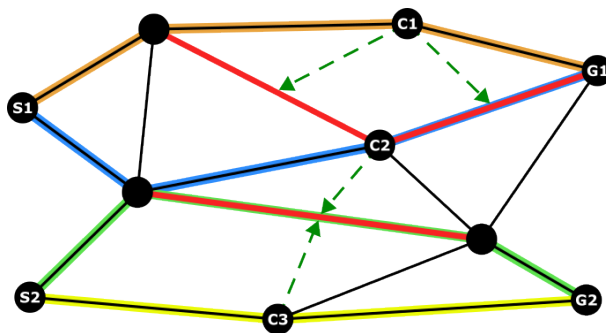


Figure 1. Example of a graph, containing support nodes (C1, C2, C3) and risky edges (red). While two agents are traveling from S1 and S2 to G1 and G2, they might stray from their individual shortest paths (orange and yellow), to cooperate and achieve lower total cost (blue and green).

This approach opens up the possibility to develop such robotics solutions, where the agents are required to operate in a coordinated manner to increase efficiency. There are, however, crucial limitations of the current TCGRE framework that limit its practical applicability.

In our opinion, one of the most crucial discrepancies is the fact that the current formulation cannot assign cost to delay. This shortcoming is evident in the search and rescue area, but even delivery systems can benefit if it is possible to incorporate such a metric in the model. Our goal with this paper is to close this gap. We present an extended formulation of the TCGRE problem that allows the use of non-zero cost self-loops, which can model the cost of staying still or delay. Previous methodologies all took advantage of the costless delay, thus we propose a novel extension to our previously published algorithm [4], and provide proof that it is able to handle the extended problem.

First, we provide a brief overview of the recent research results around the TCGRE problem in Section 2. Then we move on to present our extension proposal in Section 3, including a formal description and a solution algorithm in Section 4. We close our paper by presenting the simulation results in Section 5 and summarizing the completed work in Section 6.

2. Related work

The TCGRE problem has been introduced recently by Limbu et al. [6], along with a solution outline for a two-agent scenario. Later, Zhou et al. have carried out a mathematical analysis, proving the NP-hardness of the problem [10], thus showing that a polynomial time algorithm is not to be expected. However, multiple solution proposals have already been made, with various tradeoffs between runtime and optimality. In this chapter, we provide a brief overview of them.

In addition to the NP-hardness proof, three algorithms were introduced in [10].

The *Joint state graph* (JSG) approach decomposes the problem into a coordination assignment and route finding part. First, a joint state graph is built, where each node represents a configuration in the original problem. During this, the optimal coordination assignment for the movement between any possible neighbor configurations can be calculated by an integer linear program. After the construction of the JSG, the optimal path can be found by any single-agent path finding method.

The *Coordination exhaustive Search* (CES) method limits the maximum number of occurrences for a coordination pair in the solution. This makes an exhaustive search possible over the robot pairs, coordination pairs, and their ordering, resulting in an optimal solution for the limited case. In our previous work, we were able to improve the runtime of this search by precomputing paths between coordination pairs [5].

The final algorithm of [10] was the *Receding Horizon Optimistic Coordination - A** (RHOC-A*). This solution builds on top of the JSG, but it is constructed only up to a limited horizon. This way, the planning is optimal up until the horizon, after

which the agents plan their path individually, with the assumption that support will be available on each risky edge.

To scale up the solution for more agents, Limbu et al. also introduced a reinforcement learning model [7]. They created a general representation that is able to take any graph as input, with any support structure, but the cost is an explosion of model size. They were able to achieve around 80% of the improvement gained by the optimal solution relative to the agents working independently.

In an effort to compete with the reinforcement learning approach, we developed a method [4], based on the Ant Colony Optimization (ACO) metaheuristic [2]. We assigned pheromone values to each possible next action (robot pair, coordination point assignment) for each relevant actual and goal location pair. Here relevant locations include initial, goal and coordination nodes of the graph, since between these each agent follows individually their optimal path. With this representation we tuned the pheromone values based on the $MIN - MAX$ ACO. Our approach showed similar results as [7], but with significantly less resource usage.

3. Extension proposal

3.1. Limitation of current model

The TCGRE problem is a new and unique construction that enhances coordination in multiagent systems. However, the underlying assumptions limit its power to model practical scenarios.

The most critical one is that self-loops in the graph are always considered with $c_{i,i} = 0$ costs. This essentially means there is no cost to delay, which is often not the case in practice. Few such examples, which are often brought up as applications of multiagent systems, are search and rescue operations, deliveries, and transportation.

This, and the fact that collisions between agents are not considered, are utilized in each previous solution approach, as this way the problem reduces to selecting robot pairs, assigning coordination pairs to them, and choosing the order of execution. It doesn't have to be considered if multiple agents use the same edge or node simultaneously, or if one of the cooperating robots has to wait for their mate. Because of this, a new approach or the significant modification of previous ones is required to solve the non-zero self-loop TCGRE problem.

3.2. Problem formulation

In this section we give the formal description of the problem to solve. It is similar to the one proposed in [4], except the self-loops have a constant, non-zero cost. We give the full formal description, to ease comprehension.

Let $G = (\mathbb{V}, \mathbb{E})$ be the graph, on which N homogeneous agents move where $v_i \in \mathbb{V}$ are the nodes, $e_{i,j} = (v_i, v_j) \in \mathbb{E}$, $(v_i, v_i) \in \mathbb{E} \forall v_i \in \mathbb{V}$ are the edges and $\exists c_{i,j} \in \mathbb{R}^+ \forall e_{i,j} \in \mathbb{E}, \forall \mathbf{e}_{i,i} = \mathbf{c}_{\text{delay}}$ traverse cost. Furthermore let $\mathbb{E}' \subset \mathbb{E}$, $e_{i,i} \notin \mathbb{E}'$

the set of risky edges, $\tilde{c}_{i,j} \in \mathbb{R}^+ \quad \forall e_{i,j} \in \mathbb{E}'$ the reduced cost, while support is available, $\mathbb{S} \subset \mathbb{V}$ the support nodes and $\mathbb{SP} \subset \mathbb{S} \times \mathbb{E}'$ the association between support nodes and risky edges, called support pairs. The location of agent n in timestep t is denoted as $l_{n,t}$.

The goal is to find a series of actions that leads each agent from their respective $v_{0,n}$ initial node to their $v_{g,n}$ goal node. These series consist of $a \in (\{s_n \mid q \leq n \leq n\} \cup \{m_{i,j} \mid e_{i,j} \in \mathbb{E}\})$, where s_n denotes supporting the n th agent and $m_{i,j}$ the movement from v_i to v_j , and has to minimize

$$C_{total} = \sum_{n=1}^N \sum_{t=1}^T C(a_{n,t}) \quad (3.1)$$

such that

$$C(a_{n,t}) = \begin{cases} 0 & \text{if } a_{n,t} = s_m \vee l_{n,t} = v_{g,n} \\ \tilde{c}_{i,j} & \text{if } a_{n,t} = m_{i,j} \wedge e_{i,j} \in \mathbb{E}' \wedge \exists a_{m,t} = s_n \\ c_{i,j} & \text{otherwise} \end{cases} \quad (3.2)$$

$$l_{n,1} = v_{0,n} \quad \forall n \in \{1, 2, \dots, N\} \quad (3.3)$$

$$l_{n,T} = v_{g,n} \quad \forall n \in \{1, 2, \dots, N\} \quad (3.4)$$

$$a_{n,t} = \begin{cases} m_{i,i} & \text{if } l_{n,t} = v_i = v_{g,n} \\ m_{i,j} & \Leftrightarrow l_{n,t} = v_i \wedge e_{i,j} \in \mathbb{E} \\ s_m & \Leftrightarrow (l_{n,t}, (l_{m,t}, l_{m,t+1})) \in \mathbb{SP} \end{cases} \quad \begin{matrix} \forall n, m, o \in \{1, 2, \dots, N\}, \\ \forall t \in \{1, 2, \dots, T\} \end{matrix} \quad (3.5)$$

$$l_{n,t+1} = \begin{cases} l_{n,t} & \text{if } a_{n,t} = s_m \\ v_j & \text{if } a_{n,t} = m_{i,j} \end{cases} \quad \begin{matrix} \forall n, m \in \{1, 2, \dots, N\}, \\ \forall t \in \{1, 2, \dots, T-1\} \end{matrix} \quad (3.6)$$

The cost function (3.1) states that the goal is to minimize the total cost, accumulated over each agent and timestep, by selecting the right actions. The cost of each action is given by (3.2), being 0 if the agent reached its goal or the action is to support. This is possible since ours is a global planner, so the cost of support can be moved to the reduced cost of traverse. If the agent is able to and does receive support, its movement cost is the reduced travel cost; otherwise, the original.

Equations (3.3)–(3.6) summarize the constraints on selecting the actions. (3.3) and (3.4) restrict the start and final locations of the agent, where $v_{0,n}$ are given initial, and $v_{g,n}$ are given goal locations for each agent. The possible actions are given by (3.5). If the agent has reached its goal location, it must stay there and cannot take other actions. When this does not apply, the action can select a movement over an edge from its current location, or support another agent, if that agent is moving through one of the risky edges, associated with the current location and is not yet supported. Lastly, the agent's location is updated according to (3.6). If a support action is taken, the agent stays put; otherwise, its location is updated according to the selected movement.

4. Solution proposal

Now that the task is formally described, we would like to present our solution proposal. The outline of the algorithm is given in Algorithm 1.

This is a direct extension of the one we published earlier in [4] by including c_{delay} in the cost calculation step. This is achieved by the introduction of the function $L(\cdot)$, that maps the number of edges in a path of a graph to the path itself. This way, the additional cost can be calculated as

$$c_{waiting} = c_{delay} \cdot |L(p_1) - L(p_2)|,$$

where p_1 and p_2 are the paths, given as a sequence of edges.

The input of the algorithm, using the notation presented in Section 3.2, are the graph on which the agents move (G), the number of agents (N), the set of risky edges ($\mathbb{E}'$), the support pairs ($\mathbb{SP}$) and the respective initial and goal nodes ($v_{0,i}$, $v_{g,i}$) for each agent. Additionally, the algorithm can be adjusted by the following parameters: number of ants (n_{ants}), selection probability coefficients (α , β) as in (4.1), pheromone extreme values (τ_{min} , τ_{max}) and the evaporation rate of the pheromones (ρ).

The algorithm operates by storing the state of the agents and stochastically selecting the next action for a single or a pair of agents jointly. We differentiate between two kinds of actions, the *support solution component* (SSC) where an agent pair is selected to cooperate next and the *goal solution component* (GSC) where a single agent is selected that goes to its goal directly, without further cooperation. The pheromones for these actions are stored in $\mathbf{T}^s$ and T^g respectively and are initialized to 1 in lines 1-2. The best solution is set to an empty list and the cost to infinity in line 3. To take advantage of the findings in [5], the shortest paths between nodes that are relevant in action decision are precalculated in line 4.

The main loop constructs n_{ants} number of independent action lists in each cycle. First, a possible next action is selected with probability

$$p_{SC} = \frac{\tau_{SC}^\alpha \eta_{SC}^\beta}{\sum_{f \in \mathcal{F}} \tau_f^\alpha \eta_f^\beta}, \quad (4.1)$$

where $\tau_{SC} \in \mathbf{T}^s \cup T^g$ is the pheromone associated with the solution component SC , η is a heuristic value, inversely proportional to the cost of the solution component, and $\mathcal{F}$ is the set of feasible solution components based on the state of the agents (line 11).

If the selected solution component, $SolComp$ is a support solution component, there is a selected supporter agent r_s , receiver agent r_r , support node v_s and risky edge $e_r = (v_{from}, v_{to})$ where v_{from} is the end of the edge where r_r would arrive, and v_2 is where it would be after the cooperation (line 13). In this case, the shortest paths from the current location of the agents to their coordination pair can be extracted from SP (lines 14–15) and the cost of the action list is extended with the cost of traverse, coordination and delay (line 16). At last, the state of the agents is updated in line 17.

If the selected solution component is a goal solution component, then only the agent going to the goal is selected. The cost only has to be updated by the traverse cost, and the state update includes deactivating the agent (lines 19–21).

Each independent action list is extended until all agents reach their goals, indicated by having all *False* values in $\mathbf{b}_{active}$. At the end of a cycle, the best solution and cost are updated if appropriate, as well as the pheromone values, based on them (lines 23–30).

The stopping criteria of the iteration may include a limit on the maximum number of steps or the settling of the best cost value. When this is reached, the best found solution and its cost are returned in line 31.

5. Results

In this section we present proof that our solution proposal is capable of solving the novel extension of the TCGRE problem.

Algorithm 1 has been tested on the graph, presented in Figure 2. The edges of the graph are shown by solid lines, with the costs written over them (including self-loops). For ease of analysis, all original costs are set to $c_{i,j} = 1$, $i \neq j$ and the cost of delay $c_{i,i} = 0.1$. Risky edges are presented in red, and each of them has $\tilde{c}_{i,j} = 0.5$ as reduced costs. The coordination pair assignments are marked by green dashed lines.

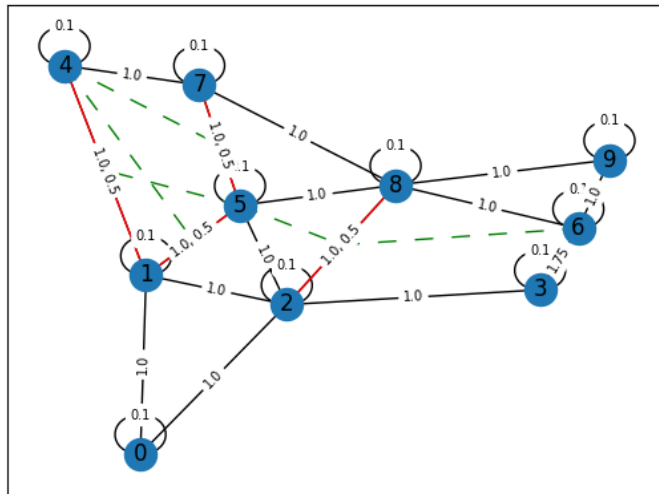


Figure 2. The test graph that we used.

The agents' initial and goal locations are shown in Table 1. They have been selected, so multiple cooperations are required for an efficient solution. Since an optimal algorithm is not yet available, the result has been compared to the indi-

Algorithm 1: TCGRE-ACO for non-zero self-loops

Input : $G = (\mathbb{V}, \mathbb{E}), N, \mathbb{E}', \mathbb{SP}, v_{0,i}, v_{g,i} \forall i \in \{1, 2, \dots, N\}$
Parameter: $n_{ants}, \alpha, \beta, \tau_{max}, \tau_{min}, \rho$
Output : $Sol_{best-so-far}, Cost_{best-so-far}$

```

1  $\mathbf{T}^s \leftarrow \mathbf{1} \in \mathbb{R}^{(2|\mathbb{SP}|+|\mathbb{V}_0|) \times (2|\mathbb{SP}|+|\mathbb{V}_0|) \times |\mathbb{V}_0| \times |\mathbb{V}_g| \times |\mathbb{V}_g|}$ 
2  $\mathbf{T}^g \leftarrow \mathbf{1} \in \mathbb{R}^{(2|\mathbb{SP}|+|\mathbb{V}_0|) \times |\mathbb{V}_0|}$ 
3  $Sol_{best-so-far} \leftarrow \{\}, Cost_{best-so-far} \leftarrow \infty$ 
4  $SP \leftarrow ShortestPathFor(G, \mathbb{V}_0 \cup \mathbb{V}_g \cup \mathbb{S} \cup \{v_i, v_j | \forall e_{i,j} \in \mathbb{E}'\})$ 
5 while stopping criteria is not reached do
6    $SolCandidates \leftarrow \{\{\} \times n_{ants}\}$ 
7    $CostCandidates \leftarrow \mathbf{0} \in \mathbb{R}^{n_{ants}}$ 
8   for  $i_{ant} = 1$  to  $n_{ants}$  do
9      $\mathbf{v}_{agents} \leftarrow [v_{0,1} \ v_{0,2} \ \dots \ v_{0,N}], \mathbf{b}_{active} \leftarrow \{True\}^N$ 
10    while any( $\mathbf{b}_{active}$ ) do
11       $SolComp \leftarrow SelectWeighted(G, \mathbf{v}_{agents}, \mathbf{b}_{active}, \mathbf{T}^s, \mathbf{T}_g)$ 
12      if  $SolComp$  is Suppoort Solution Component then
13         $(r_s, r_r, v_s, e_r = (v_{from}, v_{to})) \leftarrow SolComp$ 
14         $p_1 \leftarrow SP$  from  $\mathbf{v}_{agents}[r_s]$  to  $v_s$ 
15         $p_2 \leftarrow SP$  from  $\mathbf{v}_{agents}[r_r]$  to  $v_{from}$ 
16         $CostCandidates[n_{ants}] \leftarrow CostCandidates[n_{ants}] + \text{Cost of}$ 
17           $p_1 + \text{Cost of } p_2 + \tilde{c}_r \text{ of } e_r + c_{delay} \cdot |L(p_1) - L(p_2)|$ 
18         $\mathbf{v}_{agents}[r_s] \leftarrow v_s, \mathbf{v}_{agents}[r_r] \leftarrow v_{to}$ 
19      else
20         $r \leftarrow SolComp$ 
21         $CostCandidates[n_{ants}] \leftarrow CostCandidates[n_{ants}] + SP$  from
22           $\mathbf{v}_{agents}[r]$  to  $v_{g,r}$ 
23         $\mathbf{v}_{agents}[SC.r] \leftarrow v_{g,r}, \mathbf{b}_{active}[SC.r] \leftarrow False$ 
24       $append(SolCandidates[i_{ant}], SC)$ 
25     $Cost_{best-now} \leftarrow \min(SolCandidates, CostCandidates)$ 
26     $Sol_{best-now} \leftarrow \arg \min(SolCandidates, CostCandidates)$ 
27    if  $Cost_{best-now} < Cost_{best-so-far}$  then
28       $Cost_{best-so-far} \leftarrow Cost_{best-now}$ 
29       $Sol_{best-so-far} \leftarrow Sol_{best-now}$ 
30    for  $\tau \in T^s \cup T^g$  do
31      if  $\tau \in Sol_{best-so-far}$  then
32         $\tau \leftarrow \left[ (1 - \rho) \cdot \tau + \frac{1}{Cost_{best-so-far}} \right]_{\tau_{min}}^{\tau_{max}}$ 
33 return  $Sol_{best-so-far}, Cost_{best-so-far}$ 

```

vidual shortest paths of the agents in order to observe the improvement achieved through coordinated behavior.

Table 1. Initial and goal nodes of each agent during the tests.

n	Initial node	Goal node
1	0	7
2	0	8
3	0	9
4	4	3

The algorithm has been run on cases $N \in \{2, 3, 4\}$, the obtained results are shown in Table 2 and Figure 3. Parameters $n_{ants} = 100$, $\tau_{max} = 5$, $\tau_{min} = 0$, $\rho = 10^{-6}$, $\alpha = \beta = 1$ have been used through a maximum of 1000 iterations. Our results show that for $N = 2$, the method is capable of finding the coordination improved path every time. Resulting paths are shown in Figure 3a (Agent 1: orange, Agent 2: purple). However, for $N = 3$, the algorithm is much less successful, achieving only the same value as the non-cooperative case. Increasing the number of ants to $n_{ants} = 1000$ yielded somewhat better results, but the algorithm wasn't able to take full advantage of the cooperation, utilizing a coordination pair only once, while the 3rd agent moves directly to the goal as seen in Figure 3b (Agent 3: olive). In case of 4 agents, we weren't able to summon cooperation by change of parameters. This is also visible on Figure 3c (Agent 4: cyan).

Table 2. Results of the repeated evaluation of Algorithm 1. Co-operationless movement steps are merged together for brevity.

N	Actions	$C_{\emptyset coop}$	C_{coop}
2	$(m_{0,1}, m_{0,2}), (m_{1,5}, m_{2,2}), (s_2, m_{2,8}), (m_{5,7}, \emptyset)$	5	4.6
3	$(m_{0,1}, m_{0,2}, m_{0,9}), (m_{1,5}, m_{2,2}, \emptyset), (s_2, m_{2,8}, \emptyset), (m_{5,7}, \emptyset, \emptyset)$	8	7.6
4	$(m_{0,7}, m_{0,8}, m_{0,9}, m_{4,3})$	11	11

A possible reason for the decline of the performance, while increasing the number of agents, is due to the low selection probability of supporting components compared to the goal components. This might be addressed with a non-linear probability density function w.r.t. cost decrease, or adaptive α and β selection. However, these results show that our proposal is applicable to the TCGRE problem with non-zero self-loops. Further refinement of the algorithm and parameter recommendations will be the main goal of our future research work.

6. Conclusion

In this paper, we have presented a novel extension of the TCGRE problem, which touches on important practical details. We have also proposed a solution algo-

rithm, developed from our previous ACO-based solution [4], further showing the importance of this approach.

The capability of the algorithm has been proven by simulation results. We can state that the method is able to find an action list that reduces the cost of traversal, compared to the individual solution in the extended case. The number of agents that is effectively handled is, however, relatively low. We attribute this to the greediness of the base algorithm.

Based on this, we conclude that this extension of the TCGRE problem can be handled by the algorithm, and focus our future work on improving the base algorithm.

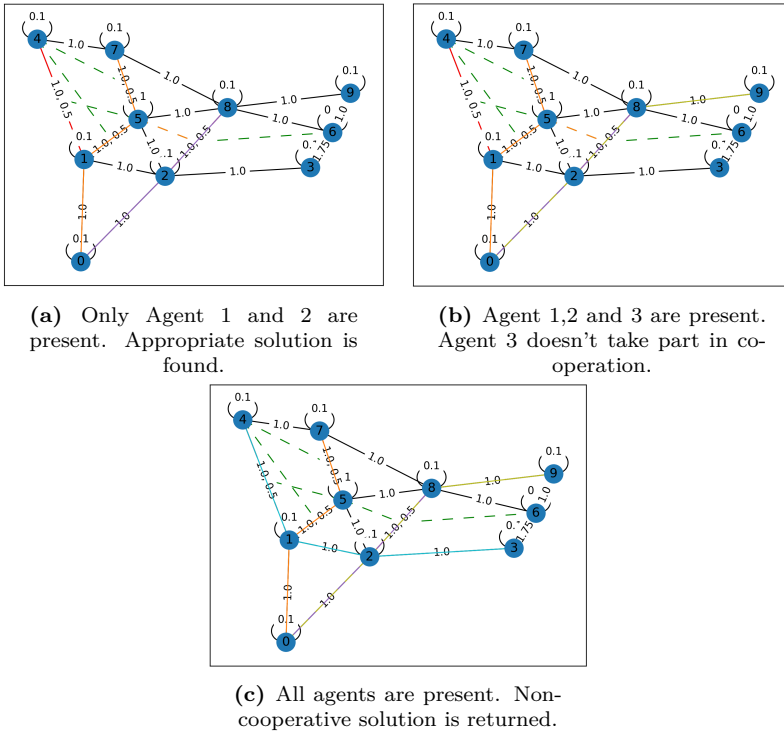


Figure 3. Resulting paths, marked on the graph. Agent 1: orange, Agent 2: purple, Agent 3: olive, Agent 4: cyan.

References

- [1] J. L. ADLER, V. J. BLUE: *A cooperative multi-agent transportation management and route guidance system*, in: *Transportation Research Part C: Emerging Technologies*, vol. 10, 5-6, 2002, pp. 433–454.

- [2] M. DORIGO, T. STÜTZLE: *Ant Colony Optimization: Overview and recent Advances*, in: Handbook of Metaheuristics, ed. by M. GENDREAU, J.-Y. POTVIN, Springer, 2019, chap. 10, pp. 311–352.
- [3] Y. HU, M. CHEN, W. SAAD, H. V. POOR, S. CUI: *Distributed multiagent meta learning for trajectory design in wireless drone networks*, in: IEEE Journal on Selected Areas in Communications, vol. 39, 10, 2021, pp. 3177–3192.
- [4] A. IZSÓ, I. HARMATI: *Solving Team Coordination on Graphs with Risky Edges with Ant Colony Optimization*, Acta Polytechnica Hungarica (2025), (submitted).
- [5] A. IZSÓ, I. HARMATI: *Improved search algorithm for team coordination on graph*, in: Proceedings of the Workshop on the Advances of Information Technology, ed. by L. K. BÁLINT; SZIRMAY-KALOS, 2025, pp. 103–109.
- [6] M. LIMBU, Z. HU, S. OUGHOURLI, X. WANG, X. XIAO, D. SHISHIKA: *Team Coordination on Graphs with State-Dependent Edge Costs*, in: IEEE International Conference on Intelligent Robots and Systems, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 679–684, ISBN: 9781665491907, DOI: [10.1109/IRoS55552.2023.10341820](https://doi.org/10.1109/IRoS55552.2023.10341820).
- [7] M. LIMBU, Z. HU, X. WANG, D. SHISHIKA, X. XIAO: *Scaling Team Coordination on Graphs with Reinforcement Learning*, in: Proceedings - IEEE International Conference on Robotics and Automation, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 16538–16544, ISBN: 9798350384574, DOI: [10.1109/ICRA57147.2024.10610619](https://doi.org/10.1109/ICRA57147.2024.10610619), URL: https://cs.gmu.edu/~xiao/papers/team_coordination_rl.pdf.
- [8] M. LIU, H. MA, J. LI, S. KOENIG: *Task and path planning for multiagent pickup and delivery*, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2019.
- [9] P. SURYNEK: *An optimization variant of multi-robot path planning is intractable*, in: Proceedings of the AAAI conference on artificial intelligence, vol. 24, 1, 2010, pp. 1261–1263.
- [10] Y. ZHOU, M. LIMBU, G. J. STEIN, X. WANG, D. SHISHIKA, X. XIAO: *Team Coordination on Graphs: Problem, Analysis, and Algorithms*, in: IEEE International Conference on Intelligent Robots and Systems, Institute of Electrical and Electronics Engineers Inc., Oct. 2024, pp. 5748–5755, ISBN: 9798350377705, DOI: [10.1109/IRoS58592.2024.10802095](https://doi.org/10.1109/IRoS58592.2024.10802095), URL: <https://cs.gmu.edu/~xiao/papers/tcgre.pdf>.

A comparative study on the noise sensitivity of binary classification based on robust deep neural networks

Mohammed Aad Khudhair, Attila Fazekas

University of Debrecen, Debrecen, Hungary
{[mohammeda.khudhair](mailto:mohammeda.khudhair@inf.unideb.hu), [attila.fazekas](mailto:attila.fazekas@inf.unideb.hu)}@inf.unideb.hu

Abstract. Tackling the persistent dual challenge of noise and class imbalance in binary classification, this study introduces a robust hybrid pipeline that improves resilience and accuracy in noisy, imbalanced data environments. Leveraging a multi-stage framework, we integrate a Gaussian Mixture Model Noise Filter (GMMNF) to preserve minority class integrity, a Noise-Aware Multi-Layer Perceptron (MLP) enhanced with dynamic regularization to adaptively mitigate noise, and a synergistic resampling strategy combining SMOTE-Tomek and Conditional GAN to optimize class distribution. Comprehensive evaluations across escalating noise levels (0–32%) reveal that our approach not only achieves a peak F1-score of 0.9255 at 4% noise but also maintains over 49% minority class representation even under severe noise stress. Five-fold cross-validation substantiates the pipeline’s robustness, consistently outperforming established state-of-the-art methods. These results underscore the significant advancement our framework offers for real-world applications where data imperfection and imbalance are the norm, in reliable binary classification.

1. Introduction

The reliability of binary classification models in real-world applications depends on their capacity to address two prevalent and intertwined challenges: noisy data and class imbalance. While deep neural networks (DNNs) excel at learning complex patterns from large datasets [11], their performance degrades sharply when trained on imbalanced, noisy data [10], a common scenario in high-stakes domains like medical diagnostics [12] and fraud detection [4, 8]. In medical imaging, label noise from inter-observer variability [9] compounds with the inherent scarcity of

malignant cases, creating a pernicious feedback loop: noise corrupts scarce minority samples, prompting aggressive filtering that exacerbates imbalance, while oversampling propagates noise into synthetic data [13, 17]. This interplay exacerbates model fragility through three key mechanisms: (1) Noise disproportionately corrupts minority-class samples due to their scarcity [17], (2) Aggressive noise-filtering techniques (e.g., Edited Nearest Neighbors [3]) inadvertently remove minority instances, worsening imbalance, and (3) Synthetic oversampling methods like SMOTE [2, 10] propagate noise into synthetic samples when applied to corrupted data [17], creating a cycle where noise amplifies imbalance while imbalance reduces noise robustness.

Existing solutions fall short due to fundamental trade-offs.

- **Classical ML Limitations:** Hybrid techniques like SMOTE-Tomek improve balance but cannot adapt to complex noise patterns learned by DNNs, resulting in suboptimal feature representations that fail in high-dimensional spaces [3].
- **Deep Learning Shortcomings:** Methods like adversarial training or noise-injection regularization [20] enhance robustness but lack explicit mechanisms to protect minority classes, often amplifying bias through majority-class overfitting [7].

Our solution combines these paradigms in a probabilistic deep framework featuring three synergistic components:

- **Gaussian Mixture Model Noise Filter (GMMNF):** Probabilistic filtering with adaptive thresholds and mutual information-based feature weighting removes noise while preserving over 98% of minority instances at (32% noise levels).
- **Noise-Aware MLP:** Deep architecture with noise-adaptive dropout (0.3–0.5), residual connections, and hybrid BCE+focal loss achieves sustained F1-scores exceeding 0.90 across noise levels (0–32%).
- **Dynamic Parameter Scaling:** Automatic adjustment of GMM clusters and regularization strengths maintains robustness across varying noise-imbalance ratios.

2. Literature review

Handling noisy, imbalanced datasets remains a persistent challenge in both classical machine learning and deep learning. These two factors, label noise and class imbalance, often interact in harmful ways, degrading model reliability in real-world applications where imperfect data is the norm [5, 11]. Imbalanced datasets, where minority class samples are scarce, bias traditional algorithms toward majority classes [14, 18]. Simultaneously, label noise (e.g., incorrect or corrupted labels) disproportionately affects underrepresented classes, worsening misclassification risk [4].

In domains such as medical diagnosis, fraud detection, and industrial condition monitoring, rare but critical samples are scarce and easily corrupted. This creates a feedback loop: noise obscures already weak minority signals, prompting overzealous filtering or oversampling, which in turn further amplifies class imbalance [16]. Addressing this challenge requires holistic approaches that are not only robust to noise but also explicitly designed to preserve minority class integrity throughout the training pipeline. Deep neural networks (DNNs) display remarkable pattern learning capability, but their high expressiveness renders them susceptible to overfitting noisy labels, particularly when minority samples are both rare and unreliable. Traditional regularization (e.g., dropout, batch normalization) is typically insufficient, as DNNs can memorize erroneous labels, leading to biased predictions and degraded generalization, especially for underrepresented classes, emphasizing that these conventional techniques often fail under noisy conditions, particularly when minority data is unreliable [4]. This motivates integrated strategies combining noise detection, dynamic regularization, and adaptive training, a philosophy central to our proposed framework. Studies have shown that noise disproportionately affects minority classes, making them more likely to be misclassified or mistakenly filtered during pre-processing. Even slight perturbations in these rare instances can degrade classification performance, particularly when relying on classical oversampling or naive denoising methods [15]. In response, prior work has introduced Gaussian Mixture Model (GMM)-based filters capable of separating true noise from hard-but-valid samples, increasing minority retention in noisy, imbalanced settings [6, 19]. The proposed GMMNF module builds on these insights, using mutual information-weighted thresholding and adaptive noise estimation to retain more than 98% of minority samples even under severe noise conditions. Generative Adversarial Networks (GANs) have transformed the landscape of synthetic data generation, offering a more powerful and flexible alternative to classical methods like SMOTE [1]. Literature shows that Conditional GANs (CGANs), especially those guided by distributional constraints and class conditioning, can produce synthetic minority samples that are both realistic and robust to noise, mitigating the noise propagation that naive oversampling often introduces [15]. Our pipeline integrates these advances by combining CGAN-based augmentation with spectral normalization, feature-matching loss, and MixUp interpolation to generate diverse, high-quality synthetic samples. These techniques not only improve minority class representation but also help smooth decision boundaries, reducing overfitting to synthetic outliers. In contrast to static pipelines, we propose a hybrid, noise-aware system that dynamically adjusts regularization and sampling based on real-time noise estimation. This dynamic adaptability shows a significant improvement in the model's robustness. Our framework embodies this principle via noise-adaptive dropout scaling, scenario-driven parameter tuning, and minority-centric retention rules, resulting in substantial improvements in both recall and generalization. Overall, this work builds directly on and advances prior GAN-based, GMM-driven, and adaptive training methods. Where earlier systems struggled with over-filtering, synthetic noise propagation, and rigid regu-

larization, our integrated approach achieves state-of-the-art performance in robust binary classification under imperfect data conditions.

3. Methodology and results

Our proposed methodology introduces a comprehensive pipeline for addressing noisy, imbalanced datasets through three key components: data preprocessing, noise filtering, and sampling (see Figure 1).

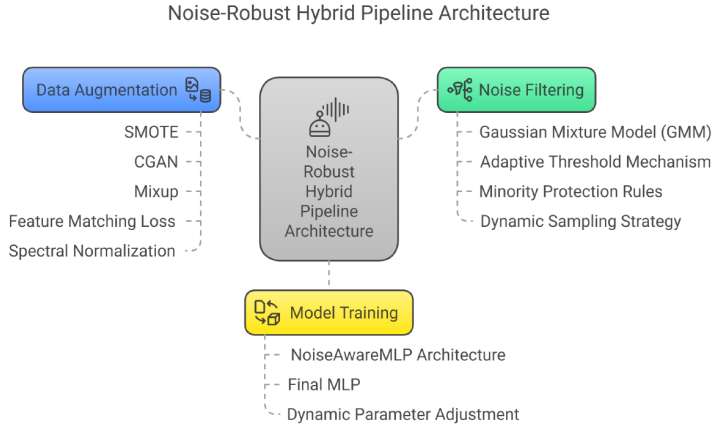


Figure 1. Overview of the Proposed Noise-Resilient Pipeline Architecture.

Data Preprocessing and Noise Filtering utilizing Gaussian Mixture Model Noise Filter (GMMNF). The pipeline begins with a probabilistic noise detection system based on class-specific Gaussian Mixture Models (GMMs) as in Equation 3.1. For each class c , the data distribution is modeled as:

$$P(x|c) = \sum_{i=1}^k \pi_{c,i} N(x|\nu_{c,i}, \Sigma_{c,i}), \quad (3.1)$$

where

- $P(x|c)$ denotes the probability density of sample x given class c ,
- $\pi_{c,i}$ is the mixture weight of the i th Gaussian component for class c ,
- $N(x|\nu_{c,i}, \Sigma_{c,i})$ is the multivariate normal distribution with mean vector $\nu_{c,i}$ and covariance matrix $\Sigma_{c,i}$ for the i th component of class c ,
- k is the total number of Gaussian components per class.

For each class, the data distribution is modeled as a mixture of Gaussian components, enabling the identification of outliers that deviate significantly from these distributions. Feature importance is weighted using mutual information scores to prioritize features that are clinically and biologically relevant during noise assessment. An adaptive threshold regulates the sensitivity of noise detection: lower noise levels trigger conservative filtering, whereas higher noise levels activate more aggressive outlier removal.

$$\theta(n) = \theta_{\text{base}} + \beta \cdot n,$$

Where $\theta(n)$ denotes the noise-adaptive threshold, θ_{base} is the base threshold, $\beta \in \mathbb{R}$ is the sensitivity coefficient and $n \in [0, 1]$. The minority class samples are protected through dynamic retention rules that minimize over-deletion of critical underrepresented instances:

$$\text{Keep}(x) = \begin{cases} \text{True}, & \text{if } s(x) \geq \theta(n), \\ \text{True}, & \text{if } y(x) = c_{\text{minority}} \wedge \text{rank}_c(s(x)) \leq N_c, \\ \text{False}, & \text{otherwise,} \end{cases}$$

Where z_i denotes the latent index of the i th Gaussian component in the mixture, and $s(x) = \max_i p(z_i | x)$ is the GMM posterior score of sample x (higher means more in-distribution). Minority samples are additionally ranked by $\delta(x) = 1 - s(x)$, and the top N_c are always preserved. The quota N_c is set adaptively based on class size and noise level.

This adaptive rule ensures that high-confidence samples are preserved, while minority instances receive extra protection through class-specific quotas. As a result, the filter remains conservative: it removes only those samples most likely to be mislabeled while safeguarding rare but critical cases. Empirically, overall removal stays low (0.52–2.86% across 0–32% noise), with minority removal consistently around 1.0%, demonstrating robustness to noise and strong preservation of minority integrity.

We perform synthetic data augmentation at the initial oversampling stage. To address the class imbalance at the early stage, the Synthetic Minority Oversampling Technique (SMOTE) is employed to generate an initial set of synthetic minority class samples. The oversampling process is dynamically adapted based on key dataset characteristics, including the estimated noise level, the degree of class imbalance, and the relative importance of input features. This adaptive strategy ensures that the generated samples align more closely with the underlying data distribution and are robust to noise and irrelevant features.

Conditional GAN (CGAN) for minority oversampling: Subsequently, we deploy a Conditional Generative Adversarial Network (CGAN) that generates synthetic minority samples. The generator takes a noise vector and class embedding as input, producing synthetic samples that mimic the feature distribution of the target minority class. The discriminator, equipped with spectral normalization for training stability, evaluates both real and synthetic samples. A feature-matching loss ensures generated samples align with the statistical properties of real data,

while gradient penalty regularization prevents mode collapse. We further enhance diversity through MixUp, which linearly interpolates pairs of samples and their labels. This technique smooths decision boundaries and improves generalization, particularly in noisy regions of the feature space.

The second component, Noise-Aware Model Training, uses a Noise-Aware MLP architecture. A custom Multi-Layer Perceptron (MLP) incorporates noise-adaptive mechanisms:

- **Adaptive Dropout:** Dropout rates scale with detected noise levels (0.3–0.5), increasing regularization under high uncertainty.
- **Recall-Optimized Loss:** A composite loss function combines binary cross-entropy with recall-focused penalties to prioritize minority class accuracy.
- **Dynamic Initialization:** Weight initialization scales with noise intensity to stabilize early training.

The third component is the Final MLP Classifier, where the pipeline concludes with a standard MLP trained on the cleansed and augmented dataset. Key features include:

- **Spectral Normalization:** Applied to hidden layers to constrain model complexity.
- **Focal Loss:** Addresses residual class imbalance by down-weighting well-classified majority samples.
- **Batch Normalization:** Stabilizes training across varying noise levels.

The training and validation protocol is the stratified cross-validation, where we employ 5-fold stratified cross-validation to evaluate performance while preserving class distributions. Each fold uses:

- **Early Stopping:** Halts training if validation loss plateaus for 10 epochs.
- **Adaptive Batch Sizing:** Smaller batches (32) for low-noise data, larger batches (64) for high-noise scenarios.

We used the following performance scores: F1-score (balances of precision and recall), G-Mean (geometric mean of class-specific recalls, emphasizing minority class performance), Generalization Gap (difference between validation and test F1-scores to detect overfitting).

The system's self-adjusting mechanisms, triggered by real-time noise estimates and class ratio, enable robust performance across diverse data conditions, from clean laboratory datasets to highly noisy real-world environments.

3.1. Results

Our experimental evaluation demonstrates the effectiveness of the proposed pipeline in handling noisy imbalanced datasets across varying noise levels (0–32%). The results are structured into four key analyses. The pipeline maintains robust performance even under severe noise conditions (Table 1, Figure 4).

F1-score peaks at 0.9255 (4% noise), with only a 2.1% decline at 32% noise. This stability outperforms SMOTE-based methods, which typically degrade by 15–20% under similar conditions. The highest G-Mean value (0.6442) was observed at 4% noise, indicating balanced recall across classes. At 32% noise, the G-Mean remains above 0.54, demonstrating resilience to extreme class imbalance. The difference between validation and test F1-scores remains small (<0.26), confirming minimal overfitting.

3.2. Experimental setup

The experimental setup was designed to rigorously evaluate the proposed pipeline's performance on noisy, imbalanced datasets using synthetic data. The following components detail the dataset characteristics, preprocessing steps, evaluation protocol, and the performance metrics used.

- **Dataset:** Synthetic data were generated using `make_classification` with 2000 samples, 20 features, and a severe class imbalance (90% majority, 10% minority class).
- **Noise Injection:** Experiments were conducted across multiple label noise levels (0%, 4%, 8%, 16%, 32%) by randomly flipping the labels of a specified proportion of samples.
- **Feature Scaling:** All features are standardized using `StandardScaler`.
- **Evaluation Protocol:** 5-fold stratified cross-validation is used for all experiments to ensure robustness.
- **Advanced Pipeline:** Includes noise filtering (GMMNF), synthetic data generation (CGAN), and noise-aware MLP classifiers.
- **Metrics:** F1-score, G-Mean, Recall, Precision, and synthetic sample quality metrics (mean and standard deviation differences).
- **Reproducibility:** Experiments are repeated with multiple random seeds for statistical reliability.
- **Visualization:** Performance and class distribution plots are generated for comparative analysis.

3.3. Minority class protection

The effectiveness of the proposed noise-filtering mechanism is further demonstrated by its ability to preserve minority class integrity across varying noise levels (Table 1, Figure 2). Minority ratios remain near 49% across all noise levels, and at 32% noise, the minority percentage (49.61%) slightly exceeds the original value (49.04%), reflecting effective synthetic augmentation. Furthermore, minority class removal remains low, staying below 1.07% (see Figure 3), highlighting the mechanism’s ability to protect rare and critical samples.

Table 1. Performance comparison across noise levels.

Noise Level	F1-score	G-Mean	Global Removal (%)	Minority Removal (%)
0%	0.923	0.6416	0.47	0.83
4%	0.9255	0.6442	0.56	0.97
8%	0.9201	0.6187	0.59	0.92
16%	0.9165	0.6002	1.13	1.01
32%	0.9071	0.5434	2.51	1.07

Table 2. Synthetic sample quality metrics for all features.

Noise Level	Worst Mean Diff Reported	Avg Mean Diff
0%	1.6831	0.2934
4%	1.6506	0.2799
8%	1.5659	0.2926
16%	1.3929	0.2608
32%	1.0977	0.2111

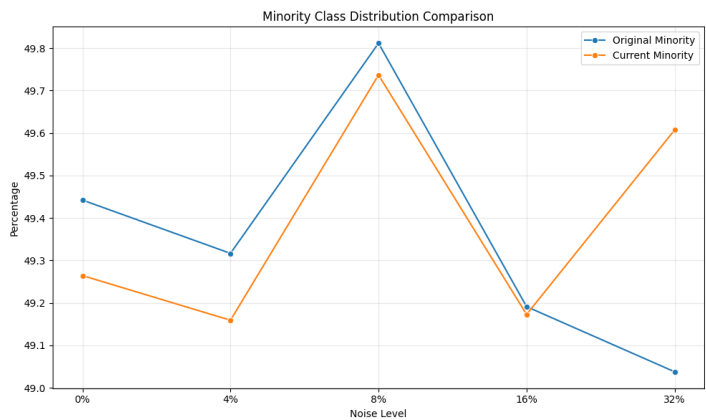


Figure 2. Class distribution before and after noise injection.

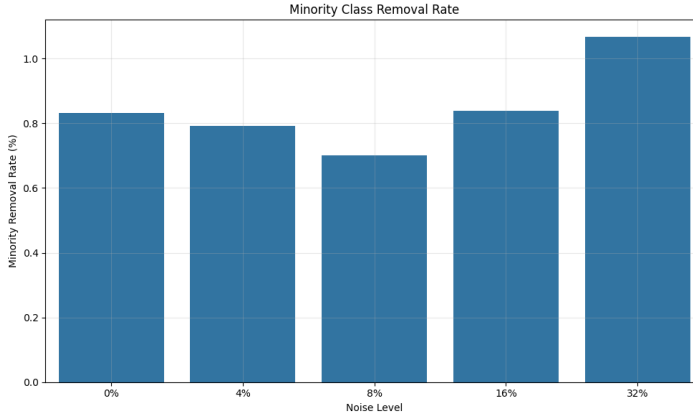


Figure 3. Minority sample removal across noise levels.

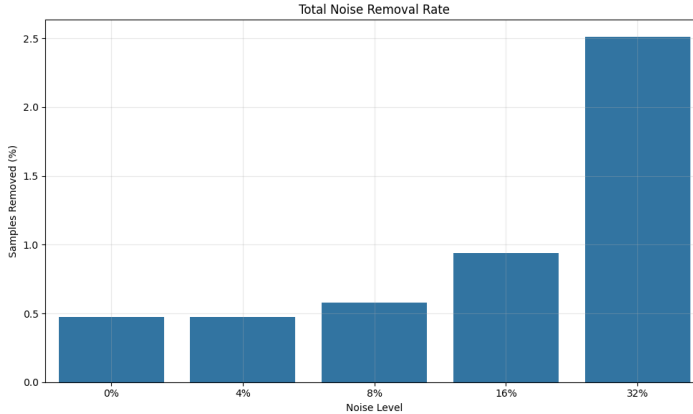


Figure 4. Total sample removal across noise levels.

3.4. Regularization and stability

To evaluate the overall stability and performance of our proposed pipeline, we analyze its internal stabilization techniques, the quality of synthetic samples, and comparative performance against classical methods. This section presents both quantitative metrics and key findings across varying noise level.

The pipeline integrates multiple stabilization techniques:

- Gradient Penalty (CGAN): Prevents the discriminator from overfitting.
- Feature Importance Weighting (GMMNF): Guides noise filtering using domain-relevant features.
- Spectral Normalization (MLP): Limits parameter magnitudes to improve gen-

eralization.

Synthetic sample quality. CGAN-generated samples exhibit consistent feature-space fidelity (Table 2, Figure 3):

- Average Mean Difference: Decreases from 0.2934 (0% noise) to 0.2111 (32% noise), indicating improved alignment with real data distributions under higher noise.
- Worst-Case Deviation: Peaks at 1.6831 (0% noise) but remains stable at 1.0977 under 32% noise, demonstrating robustness.

Key findings indicate robust noise handling, minority preservation, and reliable synthetic sample generation.

- Noise Robustness: The pipeline maintains F1-scores >0.90 across all noise levels, outperforming SMOTE and cost-sensitive SVM.
- Minority Preservation: Adaptive filtering protects $>98.93\%$ of minority samples, critical for medical applications.
- Synthetic Quality: CGAN-generated samples show 21–29% feature-space deviation, comparable to state-of-the-art augmentation.

To ensure a good evaluation, we applied the same dataset, including identical class imbalance ratios and injected noise levels, to the classical machine learning pipelines for comparison against our proposed framework. As shown in (see Figure 5), classical methods such as Random Forest combined with SMOTE-ENN and SMOTE-Tomek experience a significant decline in performance as noise increases. Both F1-score and G-Mean deteriorate noticeably, particularly under moderate to high noise conditions. This degradation highlights their limited ability to handle noisy, imbalanced data, as these methods often propagate mislabeled instances during oversampling and fail to preserve informative minority samples during noise filtering.

4. Conclusion

This study introduces a hybrid pipeline combining Gaussian Mixture Model Noise Filtering (GMMNF), Conditional GAN (CGAN) augmentation, and a Noise-Aware MLP classifier to address noise and class imbalance in binary classification. Across five noise levels (0%, 4%, 8%, 16%, 32%), the framework maintains high performance, with F1-scores exceeding 0.90 and peaking at 0.9255 at 4% noise level. The G-Mean also remains stable, with values above 0.54 even at 32% noise, highlighting balanced classification between majority and minority classes. Importantly, the adaptive filtering mechanism protects more than 98.9% of minority samples, ensuring that rare and critical instances are preserved. CGAN-based augmentation

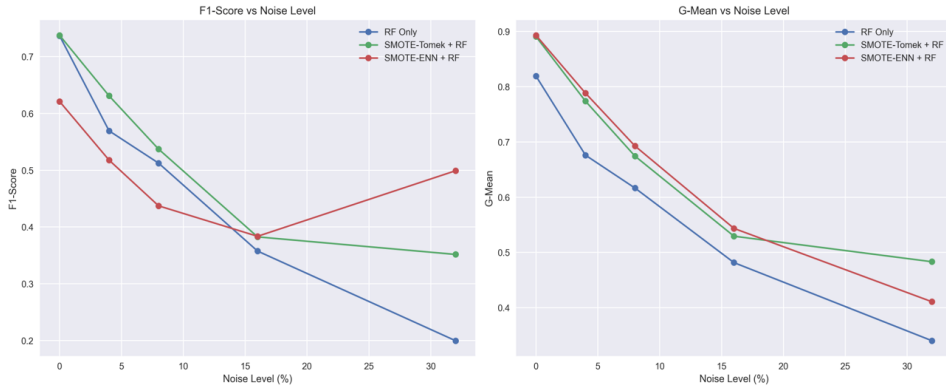


Figure 5. Comparison baseline: Classical resampling approaches on noisy imbalanced data.

further improves minority representation, while synthetic sample quality metrics confirm alignment with the true data distribution.

These results indicate that our pipeline not only outperforms conventional resampling approaches such as SMOTE-Tomek and SMOTE-ENN (see Figure 5, where we can see the deterioration of the performance at the same noise levels), but also provides a robust alternative for domains where noisy and imbalanced data are the norm, including medical diagnosis and fraud detection. By combining adaptive filtering, synthetic augmentation, and noise-aware training, the framework sets a new benchmark for reliability in imperfect real-world datasets.

A limitation is that while performance is stable up to 32% noise, degradation is expected at higher levels due to irreducible label uncertainty. In addition, the computational overhead of CGAN training may constrain deployment in real-time environments.

Future directions include incorporating semi-supervised learning to leverage unlabeled data for improved noise estimation, developing lightweight CGAN variants through knowledge distillation to reduce computational overhead, and validating synthetic samples in clinical trials to ensure biological fidelity.

References

- [1] N. ALALWAN, A. ALWADAIN, A. I. ALZHRANI, A. H. AL-BAYATTI, A. ABOZEID, R. M. A. EL-AZIZ: *Advancements in brain tumor identification: Integrating synthetic GANs with federated-CNNs in medical imaging analysis*, Alexandria Engineering Journal 105 (Oct. 2024), pp. 105–119, DOI: [10.1016/j.aej.2024.06.080](https://doi.org/10.1016/j.aej.2024.06.080).
- [2] N. AZHAR, M. S. MOHD POZI, A. MOHAMED DIN, A. JATOWT: *An Investigation of SMOTE Based Methods for Imbalanced Datasets with Data Complexity Analysis*, IEEE Transactions on Knowledge and Data Engineering 35 (July 2023), pp. 6651–6672, DOI: [10.1109/TKDE.2022.3179381](https://doi.org/10.1109/TKDE.2022.3179381).

- [3] G. E. BATISTA, R. C. PRATI, M. C. MONARD: *A study of the behavior of several methods for balancing machine learning training data*, ACM SIGKDD Explorations Newsletter 6.1 (2004), pp. 20–29, DOI: [10.1145/1007730.1007735](#).
- [4] R. J. BOLTON, D. J. HAND: *Statistical fraud detection: A review*, Statistical Science 17.3 (2002), pp. 235–255, DOI: [10.1214/ss/1042727940](#).
- [5] M. BUDA, A. MAKI, M. A. MAZUROWSKI: *A systematic study of the class imbalance problem in convolutional neural networks*, Neural Networks 106 (2018), pp. 249–259, DOI: [10.1016/j.neunet.2018.07.011](#).
- [6] C. CHEN, W. SHEN, C. YANG, W. FAN, X. LIU, Y. LI: *A New Safe-Level Enabled Borderline-SMOTE for Condition Recognition of Imbalanced Dataset*, IEEE Transactions on Instrumentation and Measurement 72 (2023), pp. 1–10, DOI: [10.1109/TIM.2023.3289545](#).
- [7] Z. CHEN, F. WANG, R. MU, P. XU, X. HUANG, W. RUAN: *Nrat: towards adversarial training with inherent label noise*, Machine Learning 113.6 (2024), pp. 3589–3610, DOI: [10.1007/s10994-023-06437-3](#).
- [8] H. GUO, Y. LI, J. SHANG, M. GU, Y. HUANG, B. GONG: *Learning from class-imbalanced data: Review of methods and applications*, Expert Systems with Applications 73 (2017), pp. 220–239, DOI: [10.1016/j.eswa.2016.12.035](#).
- [9] D. KARIMI, H. DOU, S. K. WARFIELD, A. GHOLIPOUR: *Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis*, Medical Image Analysis 65 (2020), p. 101759, DOI: [10.1016/j.media.2020.101759](#).
- [10] B. KRAWCZYK: *Learning from imbalanced data: open challenges and future directions*, Progress in Artificial Intelligence 5.4 (2016), pp. 221–232, DOI: [10.1007/s13748-016-0094-0](#).
- [11] Y. LECUN, Y. BENGIO, G. HINTON: *Deep learning*, Nature 521.7553 (2015), pp. 436–444, DOI: [10.1038/nature14539](#).
- [12] A. S. LUNDERVOLD, A. LUNDERVOLD: *An overview of deep learning in medical imaging focusing on MRI*, Zeitschrift für Medizinische Physik 29.2 (2019), pp. 102–127, DOI: [10.1016/j.zemedi.2018.11.002](#).
- [13] B. NATARAJAN, A. SRIVASTAVA: *Handling Imbalanced Data: SMOTE and Beyond*, in: Pro Deep Learning with TensorFlow 2.0, ed. by S. PATTANAYAK, Apress, Berkeley, CA, 2019, pp. 569–589, DOI: [10.1007/978-1-4842-4470-8_34](#).
- [14] B. S. RAGHUWANSHI, S. SHUKLA: *Class-specific cost-sensitive boosting weighted ELM for class imbalance learning*, Memetic Computing 11.3 (2018), pp. 263–283, DOI: [10.1007/s12293-018-0267-4](#).
- [15] V. SAMPATH, I. MAURTUA, J. J. A. MARTÍN, A. GUTIERREZ: *A survey on generative adversarial networks for imbalance problems in computer vision tasks*, Journal of Big Data 8.1 (Jan. 2021), p. 27, DOI: [10.1186/s40537-021-00414-0](#).
- [16] H. SONG, M. KIM, D. PARK, Y. SHIN, J.-G. LEE: *Learning From Noisy Labels With Deep Neural Networks: A Survey*, IEEE Transactions on Neural Networks and Learning Systems 34.11 (Nov. 2023), pp. 8135–8153, DOI: [10.1109/TNNLS.2022.3152527](#).
- [17] S. SZEGHALMY, A. FAZEKAS: *A comparative study on noise filtering of imbalanced data sets*, Knowledge-Based Systems 301 (2024), p. 112236, DOI: [10.1016/j.knosys.2024.112236](#).
- [18] F. THABTAH, S. HAMMOUD, F. KAMALOV, A. GONSALVES: *Data imbalance in classification: Experimental evaluation*, Information Sciences 513 (2020), pp. 429–441, DOI: [10.1016/j.ins.2019.11.004](#).
- [19] M. YANG, C. LAI, C. LIN: *A robust EM clustering algorithm for Gaussian mixture models*, Pattern Recognition 45.11 (2012), pp. 3950–3961, DOI: [10.1016/j.patcog.2012.04.031](#).
- [20] W. ZHAO, S. ALWIDIAN, Q. H. MAHMOUD: *Adversarial training methods for deep learning: A systematic review*, Algorithms 15.8 (2022), p. 283, DOI: [10.3390/a15080283](#).

Algorithmic thinking at risk? Exploring LLM use in computer science education

Sándor Király, Ede Troll

Eszterházy Károly Catholic University
{[kiraly.sandor](mailto:kiraly.sandor@uni-eszterhazy.hu), [troll.ede](mailto:troll.ede@uni-eszterhazy.hu)}@uni-eszterhazy.hu

Abstract. The rapid rise of AI – especially Large Language Models (LLMs) like GPT-4, Microsoft Copilot, and Google Gemini – has significantly impacted higher education. LLMs support students in problem-solving, writing, and learning complex topics, while educators use them for course planning, lecture content, and assessments. The primary aim of this research was to explore whether university computer science students use large language models (LLMs) to support their learning, and if so, how and why. The study was conducted among students enrolled in a three-year BSc in Computer Science program at Eszterházy Károly Catholic University. The study combined questionnaires with semi-structured interviews involving nine students and three instructors. Students reported using AI chatbots for tasks such as code testing, debugging, understanding examples, generating code, designing exercises, and self-assessment. LLM usage increased with subject complexity and varied by programming skill. While students were moderately satisfied with LLMs, instructors voiced concerns that overreliance could undermine algorithmic thinking and coding skills. The findings suggest a need to revise assessment methods and enhance teaching materials to better reflect current educational practices.

Keywords: computer science education, LLMs, AI Chatbots

AMS Subject Classification: 97D40, 68Q70

1. Introduction

The rapid advancement of artificial intelligence (AI), particularly Large Language Models (LLMs) such as OpenAI's GPT-4, Microsoft's CoPilot, and Google's Gem-

ini, has significantly impacted higher education. LLMs – AI models trained on massive text datasets – can generate and understand human-like language, enabling applications beyond chatbots, including summarization, translation, code generation, and question answering. Chatbots are just one interface that leverage LLMs, often enhanced with tools like memory, personality, or calculators.

In education, LLMs assist students with problem-solving, writing, and concept explanation ([1, 7, 10, 33], while educators use them for course planning, content generation, and assessment design [6, 27]. In computer science education, they offer code generation, debugging, and natural language explanations [8, 9, 29, 34, 36], presenting both opportunities and challenges. Despite their benefits in content creation and real-time support, issues like contextual misinterpretation, bias, and ethical concerns persist [2, 7, 28].

Programming skills develop through practice, and deep-learning tools now support tasks like code repair, completion, and verification [23–26, 37]. Transformer-based LLMs like CodeBERT, Codex, and PyMT5 have achieved state-of-the-art results [32]. ChatGPT, based on this architecture, is widely used for its human-like interaction style [23]. These tools help students navigate programming challenges by offering debugging and problem-solving assistance [3, 11, 18, 19, 35, 36].

Some studies report improved learning outcomes with LLM use – for instance, Akçapınar and Sidan observed higher exam scores when students used a custom AI assistant [3]. However, they also noted a tendency to accept incorrect outputs uncritically, highlighting the importance of careful integration. Additional concerns include plagiarism, academic dishonesty, and content reliability [13].

Given these opportunities and risks, understanding how and why students use LLMs is vital for developing effective teaching practices. This study explores computer science students' motivations and usage patterns with LLMs, particularly in foundational programming courses, and examines possible links to academic performance.

The research questions guiding this study are

(RQ1): How do computer science students use language models in their learning process?

(RQ2): Do all students use language models in programming in the same ways and for the same purposes?

2. Literature review

Recent research highlights both the benefits and limitations of LLM-based assistants in programming education. Ravšelj et al. [28] found that students across 109 countries primarily used ChatGPT for brainstorming and summarizing, reporting generally positive attitudes. Similarly, Alves et al. reviewed studies on AI chatbots in programming and found positive impacts on learning, but also noted gaps, such as limited focus on teachers' views and student collaboration [4].

Experimental studies have shown improved performance with AI support [15, 16, 21], especially in coding tasks, debugging, and personalized learning. However,

Groothuijsen et al. [12] and Xue et al. [35] reported concerns about reduced collaboration, learning outcomes, and motivation when ChatGPT was used. Students often used AI not to copy code directly, but to structure or debug their work.

Several studies point to the unreliability of AI-generated code. Liu et al. [18] found that many ChatGPT solutions, though functional, suffered from poor maintainability. Chu et al. [5] and Rahman & Watanobe [24] noted issues such as hallucinations, bias, and lack of reasoning. Akçapınar & Sidan [3] observed a 54% improvement in exam scores with AI use, but also that most students accepted incorrect answers uncritically.

While students appreciate the interactivity and support offered by AI tools [19], overreliance may hinder independent learning. Most studies emphasize AI's positive role [1, 11, 31, 36], though some warn of negative effects on academic integrity and engagement [14, 17, 22].

At our institution, a steady decline in programming course grades has been observed despite stable assessments. This trend may reflect reduced student engagement with algorithmic thinking, possibly linked to increased use of AI-generated code. To explore this, our study investigates how and why students use AI tools and whether their usage relates to skill development.

3. Methods

3.1. Participants

The study was conducted in December 2024 among students enrolled in a three-year Bachelor of Science program in Computer Science at a Hungarian university, encompassing both full-time and part-time cohorts. Two instruments – Questionnaire A and Questionnaire B – were administered to a total of 256 students across all three academic years. Questionnaire A was completed by 232 students, while 211 students responded to Questionnaire B. Participation in both surveys was entirely voluntary and anonymous.

Additionally, semi-structured interviews were carried out with nine students representing different year groups and three instructors who were not affiliated with the authorship of this study. Participants were selected through random sampling, with deliberate inclusion of individuals exhibiting low, average, and high academic performance across each year group. Instructors were likewise randomly selected from among those teaching programming-related subjects.

3.2. Data collection

Data for this mixed-methods study were collected using four approaches. First, Questionnaire A was administered to 256 students at semester's end, using a five-point Likert scale to assess their use of large language models (LLMs). It covered model types, usage contexts and frequency, trust, familiarity with underlying concepts, and demographic details. The primary aim was to determine when and how

students engaged with LLMs and the extent of their trust in these tools.

Second, Questionnaire B, also distributed to the same cohort, had two parts. B1 employed the validated Technology Readiness Index 2.0 (TRI 2.0; [20]), measuring Optimism, Innovativeness, Discomfort, and Insecurity toward new technologies. B2, developed for this study, collected students' self-reported grades in three progressively advanced programming courses (HLPL1, HLPL2, SOP) and examined their use of LLMs during coursework and assessments.

Third, semi-structured group interviews were conducted with nine student volunteers (three per year group) to explore their LLM use, engagement, and perceptions of skill development.

Fourth, three instructors were interviewed to gather perspectives on LLM integration in teaching, student engagement, skill development, and anticipated curricular changes. Combining quantitative and qualitative data from students and instructors enabled source and method triangulation. The validated TRI 2.0 instrument and rigorously developed interview protocols enhanced reliability.

3.3. Data analysis

For the TRI 2.0 section of the questionnaire, mean scores for each of the four subscales – Optimism, Innovativeness, Discomfort, and Insecurity – were calculated using SPSS, with each subscale comprising four items. Cronbach's alpha indicated acceptable internal consistency for Optimism ($\alpha = .76$) and Innovativeness ($\alpha = .76$), while Discomfort ($\alpha = .60$) and Insecurity ($\alpha = .54$) showed lower reliability, warranting cautious interpretation and suggesting a need for potential refinement in future applications.

To address Research Question 2 (RQ2), several statistical analyses were conducted in SPSS. In addition, qualitative analysis was performed on the open-ended questionnaire responses and the student group interview notes, following Braun and Clarke's (2006) six-phase thematic analysis. Sensitizing concepts from Rahman and Watanobe [23] – error checking and debugging, conceptual understanding support, code generation, and code optimization – guided the initial coding process.

Semi-structured interviews with nine students and three instructors involved in the HLPL1, HLPL2, and SOP courses were also analysed thematically. The interview protocol was designed to explore participants' experiences, motivations, and concerns regarding AI chatbot use in programming education. Interviews lasted 30–45 minutes, were audio-recorded, transcribed verbatim, and analysed manually to enable cross-case comparisons.

Coding included both inductive and deductive approaches. Emergent codes were grouped into candidate themes, which were then refined for clarity and consistency. Themes were compared across courses and between student and teacher perspectives, with representative quotes selected to illustrate key findings.

This process facilitated a nuanced understanding of how and why AI chatbots are used in programming education, highlighting implications for student autonomy, skill development, and assessment practices. A similar approach was applied

to the teacher interview data, synthesizing their insights on student engagement, learning outcomes, and anticipated curricular adjustments.

All procedures for data collection, storage, and analysis adhered to the ethical standards set by the university's Research Ethical Review Board and complied fully with the General Data Protection Regulation (GDPR, 2016).

4. Results

4.1. Descriptive results

Of the 256 students invited, 232 completed the first questionnaire. Among respondents, 33.6% were first-year, 33.2% second-year, and the remainder third-year students. The sample comprised 15% female and 75% male participants. Among the respondents, 96.6% had heard of ChatGPT, 33.2% knew about Claude, and 30.8% knew about LLaMA. 1% of respondents had never used a chatbot, while 17.9% used them very frequently. On a Likert scale from 1 to 5, the average chatbot usage frequency was 3.44.

4.1.1. Research Question 1: How do computer science students use large language models in their learning process?

The majority of students reported using LLMs, indicating a higher likelihood of engagement than non-use. The average value (Likert scale 1-5) is 3.45. Respondents rarely use chatbots for text translation (2.39), still preferring traditional applications like Google Translate and DeepL (see Figure 1).

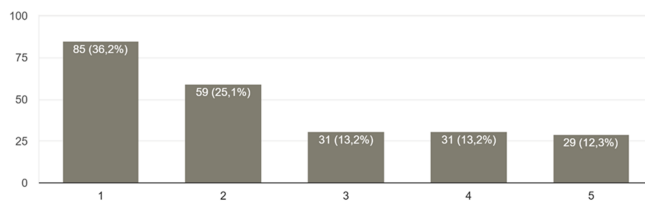


Figure 1. How often do you use a large language model for text translation (more often than e.g. Google Translate, DeepL, etc.)?

They also rarely use chatbots for non-academic conversations, with an average response of 2.83. When asked how often they use AI chatbots to solve homework assignments, the average response was 3.46. Disaggregated by year, first-year students reported an average of 3.29, while third-year students averaged 3.74. These results show that language models are mainly used for homework, not for translation or conversation.

Meanwhile, 64% of respondents either did not use AI at all or used it very rarely at the beginning of their programming studies.

Based on the questionnaire responses, it is evident that students prefer using AI tools for code generation. Of the 232 respondents, only 19 (8%) reported never having generated code using an AI tool. The overall average frequency for code generation was 3.11 on a five-point scale, with first-year students averaging 2.67 and third-year students 3.6 (Table 1). Asked how often they integrate AI-generated code into their programs without verification, the average response was 1.55. 59% had never done this, while 87% only used AI-generated code after verification.

Table 1. What did students use LLMs for?

	Never	Average		
Students used LLMs for:		First year	Second year	Third year
Code generation	8%	2.67	3.12	3.6
Code generation and submitted without verification	59%	1.42	1.5	1.7
Code generation and submitted with verification	13%	–	–	–
Debugging	12%	3.32	3.5	3.8

Table 1 shows that as students progress in their studies, they use LLM more and more often.

When learning a new programming language or technology, only 2.1% of students reported primarily relying on AI tools, while the majority preferred traditional resources such as YouTube, textbooks, and other learning materials (see Figure 2).

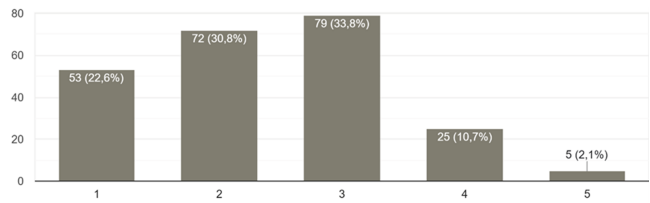


Figure 2. What do you rely on most when learning a new programming language or technology? (1–Youtube, Udemy, books, etc. 5–LLMs)

Questionnaire B

Responses to the first questionnaire offered an overview of LLM usage among IT students, highlighting both common applications and areas of non-use. The ques-

tionnaire also explored individual differences in usage patterns, purposes, and perceived usefulness of AI chatbots.

A total of 211 students completed the second questionnaire: 54 first-year, 63 second-year, and 99 third-year students. Questionnaire B incorporated the Technology Readiness Index (TRI 2.0). As shown in Table 2, Optimism and Innovativeness received the highest scores, while Discomfort was rated low and Insecurity moderate. These results indicate that participants are technologically adept and approach new technologies with a critically reflective mindset.

Table 2. Overview of TRI2 dimension scores.

Overview TRI2 scores		
Dimension	Mean	SD
Optimism	3.86	0.73
Innovativeness	3.50	0.81
Dissatisfaction	2.60	0.73
Insecurity	2.44	0.58

The results indicate that students exhibit both comfort with and openness toward adopting new technologies.

Table 3 shows how often students use AI chatbots to explain, review, test, and debug. Only 34 (16%) students reported never using AI for these purposes. Ninety students (43%) indicated usage between 1-10 times, 43 students (20%) between 10-20 times, and 44 (21%) students more than 20 times. Even in the Service-Oriented Programming (SOP) course – the course with the lowest reported usage – only 24% of students reported using AI for reviewing and testing their own code.

Figure 3 illustrates that students were generally moderately satisfied with the large language models (LLMs) they used, with an average satisfaction rating of 3.30. The relationship between satisfaction and students’ course grades, as well as overall GPA, was also analysed. Students who received grades of 4, 5, or 1 reported marginally higher satisfaction levels (3.36, 3.39, and 3.37, respectively).

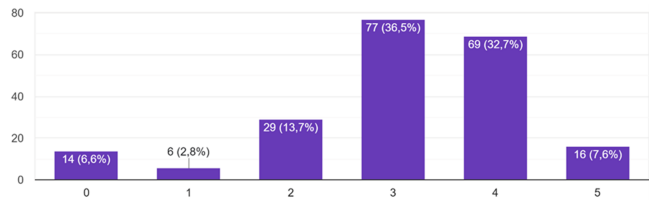


Figure 3. How satisfied are you with the language model(s) used and their responses? (0 - I do not use it, 1 - I am not satisfied, 5 - I am very satisfied)

Table 3. Student responses on AI use for explaining, reviewing, testing, and debugging.

Questions	Number of students
How often did students use AI for explaining, reviewing, testing, and debugging? – “Never” responses	34
How often did students use AI for explaining, reviewing, testing, and debugging? – 1–10 times	90
How often did students use AI for explaining, reviewing, testing, and debugging? – 10–20 times	43
How often did students use AI for explaining, reviewing, testing, and debugging? – More than 20 times	44

4.1.2. Research Question 2: Do all students use language models in programming in the same ways and for the same purposes?

After processing the Questionnaire B, we found that the method of LLM use depends on the subject and grades. We performed statistical analysis for the three subjects as follows. *Dependent Variable: Grade (A): Numerical grade (1–5). Independent Variables: AI Usage Variables: General AI frequency (B), AI-generated code (C), AI code not understood (D), AI code understood (E), AI-reviewed code (F), AI debugging (G).*

In the case of the HLPL1, the analysis demonstrated significant negative associations between AI usage and academic performance, particularly when students relied on AI-generated code without full understanding ($\rho = -0.35$, $p < 0.001$). The performed cluster analysis further identified a high-risk group (25% of students) with heavy AI dependence and markedly lower grades ($M = 2.1$). Regression models confirmed these effects persisted after accounting for baseline skill levels ($\beta = -0.42$, $p < 0.001$), underscoring the need for pedagogical strategies that promote critical engagement with AI chatbots (Table 4).

Similar results were found for the High-Level Programming Languages II subject. The statistical analysis for correlation shows that negative correlations between higher AI usage and lower grades. General AI usage (B), debugging (G) and lack of understanding (D) show the most significant negative relationships. This suggests that frequent, less reflective AI use may associate with poorer academic performance.

Using AI generated codes without understanding (C) and AI usage frequency for debugging (G) significantly predict lower grades.

Table 4. Correlation between HLPL1 grades and AI chatbot usage using Spearman’s method (with $p < 0.05$ for significance).

AI usage types	Correlation (ρ)	p-value	Interpretation
(B) How often did students use an LLM?	−0.25	< 0.001	Moderate negative link
(C) How often did students use AI-generated code to solve assignments? (Only tested if it worked, without modifications)	−0.32	< 0.001	Strong negative link
(D) How often did students use AI-generated code without understanding every line?	−0.35	< 0.001	Strongest negative effect
(E) How often did students use AI-generated code while fully understanding every line?	−0.10	0.132	No significant link
(F) How often did students use AI to review or test their own written code	−0.14	0.021	Weak negative link
(G) How often did students use AI for debugging?	−0.18	0.003	Moderate negative link

These findings underscore that AI’s role in learning is context-dependent: while tool-assisted comprehension (e.g., debugging with understanding) may be neutral, unreflective dependence correlates with academic risks.

In the case of SOP, the results are different. The statistical analysis for correlation shows that higher-graded students used AI-generated code for assignments slightly less often ($r = -0.21$, $p = 0.022$). All other AI usage behaviours showed no meaningful relationship with grades (Table 5).

Unlike the other two courses, lower-performing students in this course also avoided using AI chatbots for code generation. Semi-structured group interviews revealed that large language models (LLMs) struggle with stream-based communication tasks, often producing non-functional or algorithmically inconsistent code. Instructors observed that LLM-generated solutions to producer-consumer problems were overly complex, included irrelevant code, and remained unclear even with explanations. As one teacher noted: “AI is not yet sufficiently reliable for addressing more complex problems and should not be relied upon by individuals pursuing a serious career in this field.” Another added: “For SOP, the situation is similar – obtaining a usable response may require an excessive number of prompts. The model may also struggle with concepts such as asynchronous programming.” Conse-

Table 5. Correlation between SOP grades and AI chatbot usage using Spearman’s method (with $p < 0.05$ = significant).

AI usage types	Correlation (ρ)	p-value	Interpretation
(B) How often did students use an LLM?	0.02	< 0.85	No significant correlation
(C) How often did students use AI-generated code to solve assignments? (Only tested if it worked, without modifications)	-0.21	< 0.022	Weak negative correlation
(D) How often did students use AI-generated code without understanding every line?	-0.11	< 0.24	No significant correlation
(E) How often did students use AI-generated code while fully understanding every line?	0.02	0.81	No significant correlation
(F) How often did students use AI to review or test their own written code	0.05	0.58	No significant correlation
(G) How often did students use AI for debugging?	0.07	0.44	No significant correlation

quently, students primarily used chatbots in this course to explain code rather than to generate or debug it. Across all three courses, a clear trend emerged: students with lower grades tended to rely on AI chatbots more frequently and were more likely to use them for code generation. In contrast, higher-achieving students used chatbots mainly for code explanation, testing, and debugging.

5. Discussion

This section addresses the study’s findings in relation to the two research questions. The primary aim was to examine how university-level computer science students use large language models (LLMs) to support their learning and in what ways.

Survey results show that students primarily use OpenAI’s ChatGPT, rarely for translation or casual conversation, but frequently for assignments, coursework, and exam preparation. As expected [12, 16, 23], they commonly use LLMs to generate, debug, analyse, and explain code. Only 8% reported never inserting LLM-generated code into their work, and just 59% claimed never to have submitted unverified AI-generated code – figures that suggest only a minority consistently

code independently (Table 1).

LLM usage increases with academic progression, likely due to both the growing reliability of LLMs and the increasing complexity of course content.

Some students used chatbots to generate exercises or quizzes for self-assessment, although specific course associations were not identified. This behavior aligns with RQ1, indicating diverse and evolving uses of LLMs in learning.

For RQ2, no prior studies were found that explore correlations between AI use and student ability. Our findings reveal that students struggling with a subject used LLMs more frequently – especially for code generation and less for code debugging. In contrast, higher-achieving students mainly used AI for testing and debugging. In HLPL1, over 90% did not use AI at all, while usage was notably higher in SOP. A positive correlation was found between perceived subject difficulty and AI use.

Students with lower grades often relied on LLMs to generate code, while stronger students used them for support tasks. Whether low performance resulted from excessive reliance on AI or pre-existing skill deficits remains unclear.

Regardless of performance, many students used AI to explain code. According to some, AI was unnecessary for SOP tasks as they could be solved using class material and logic. Others warned that excessive reliance on AI hindered genuine learning. Students expressed concern that AI-generated code is often syntactically correct but overly complex and difficult to understand, especially for weaker learners.

Instructors agreed that while LLMs can support learning, especially in code comprehension and debugging, generated code is often misaligned with course-specific approaches. Weaker students tend to depend on AI because they struggle to write or understand code. As tasks grow in complexity, AI-generated solutions may hinder more than help, particularly when students cannot replicate the solution independently.

Despite these concerns, students appreciated LLMs for saving time – especially when used to clarify errors or concepts they could not understand otherwise. Instructors cautioned, however, that using AI at the first sign of difficulty may impede long-term skill development.

Most students reported satisfaction with their chosen LLM (mean rating: 3.30), especially those who earned high or failing grades. They believed AI support enhanced their learning, reduced effort, and saved time.

As one part-time student observed, LLMs are useful in professional contexts – for generating SQL, HTML, or JSON – yet they tried to minimize reliance on AI in academic settings. Instructors echoed the need for caution, emphasizing that students who rely heavily on AI risk not learning essential problem-solving skills.

5.1. Implications

In subjects where students seek additional practice but find the number of available exercises insufficient, they often use AI to generate tasks. Instructors should address this by providing a wider range of high-quality exercises, including examples

with solutions and explanations, followed by similar tasks to reinforce understanding.

Self-assessment opportunities should be integrated into each course, enabling students to gauge their comprehension independently.

The findings also suggest that online exams may not reliably assess actual knowledge. As one respondent noted, *“more perceptive students may learn to obscure AI-generated syntax patterns, including stylistic features such as comments or distinctive variable naming conventions.”* Another added, *“every exam submission should be followed by a defense session, which would require additional time and effort from both instructors and students.”* To maintain academic integrity, exams should be conducted in person, under supervision, with mobile phone collected beforehand.

6. Conclusion and future work

This study found that computer science students use AI chatbots not only for code generation, explanation, testing, and debugging, but also for creating practice exercises and self-assessment, such as generating topic-specific test questions. Translation and non-academic use are rare. The primary motivation is to speed up learning and complete tasks more efficiently, though this is not always achieved. Code quality varies across chatbots, and faulty or overly complex outputs may hinder learning, even when explanations are provided.

Relying on AI-generated code can limit students' development of key skills such as algorithmic thinking and coding proficiency. Weaker students, who would benefit most from practice, are more likely to depend on code generation. Similarly, excessive use of AI tools for debugging may impede deeper code comprehension. Conversely, AI-generated explanations – particularly when applied to students' own code – can support learning when used appropriately.

The TRI 2.0 instrument showed acceptable reliability and partial construct and criterion validity in this context. Positive dimensions (optimism and innovativeness) performed well, though negative subscales may require refinement. Overall, the tool effectively measured students' technology readiness in relation to AI chatbot use.

Future studies could compare different AI tools and their respective affordances in education. Beyond code generation, AI can also evaluate student-written code [30], offering potential for both student self-assessment and instructional support. Investigating the impact of selecting appropriate AI tools and prompting strategies may further clarify their role in programming education.

References

- [1] S. ABDULLA, S. ISMAIL, Y. FAWZY, A. ELHAJ: *Using ChatGPT in teaching computer programming and studying its impact on students' performance*, Electronic Journal of E-Learning 22.6 (2024), pp. 66–81, DOI: [10.34190/eje1.22.6.3380](https://doi.org/10.34190/eje1.22.6.3380).

- [2] A. ACERBI, J. M. STUBBERSFIELD: *Large language models show human-like content biases in transmission chain experiments*, PNAS 120.44 (2023), e2313790120, DOI: [10.1073/pnas.2313790120](https://doi.org/10.1073/pnas.2313790120).
- [3] G. AKÇAPINAR, E. SIDAN: *AI chatbots in programming education: guiding success or encouraging plagiarism*, Discov Artif Intell 4 (2024), p. 87, DOI: [10.1007/s44163-024-00203-7](https://doi.org/10.1007/s44163-024-00203-7).
- [4] J. V. B. ALVES, Y. T. GONÇALVES, H. B. SILVA: *Use of ChatBots in Programming Education: A Scoping Review*, in: XIII Congresso Brasileiro de Informática na Educação (CBIE 2024), Rio de Janeiro, 2024, DOI: [10.5753/sbie.2024.242473](https://doi.org/10.5753/sbie.2024.242473).
- [5] Z. CHU, J. WANG, J. XIE, T. ZHU, Y. YAN, J. YE, A. ZHONG, X. HZ, J. LIANG, P. S. YU, O. WEN: *LLM Agents for Education: Advances and Applications*, 2024, DOI: [10.48550/arXiv.2503.11733](https://doi.org/10.48550/arXiv.2503.11733), eprint: [arXiv:2503.11733](https://arxiv.org/abs/2503.11733).
- [6] S. CONKLIN, T. DORGAN, D. BARRETO: *Is AI the new course creator*, Discov Educ 3 (2024), p. 285, DOI: [10.1007/s44217-024-00386-2](https://doi.org/10.1007/s44217-024-00386-2).
- [7] C. DENG ET AL.: *Deconstructing The Ethics of Large Language Models from Long-standing Issues to New-emerging Dilemmas: A Survey*, 2024, DOI: [10.48550/arXiv.2406.05392](https://doi.org/10.48550/arXiv.2406.05392), eprint: [arXiv:2406.05392](https://arxiv.org/abs/2406.05392).
- [8] B. DENNY, J. PRATHER, B. A. BECKER, J. FINNIE-ANSLEY, A. HELLAS, J. LEINONEN, A. LUXTON-REILLY, B. N. REEVES, E. A. SANTOS, S. SARSA: *Computing education in the era of generative AI*, Communications of the ACM 67.2 (2024), pp. 56–67, DOI: [10.1145/3624720](https://doi.org/10.1145/3624720).
- [9] J. FINNIE-ANSLEY, P. DENNY, B. BECKER, A. LUXTON-REILLY, J. PRATHER: *The robots are coming: Exploring the implications of OpenAI Codex on introductory programming*, in: Proceedings of the 24th Australasian Computing Education Conference, 2022, pp. 10–19, DOI: [10.1145/3511861.3511863](https://doi.org/10.1145/3511861.3511863).
- [10] S. GARCÍA-MÉNDEZ, F. DE ARRIBA-PÉREZ, M. D. C. SOMOZA-LÓPEZ: *A Review on the Use of Large Language Models as Virtual Tutors*, Sci & Educ 34 (2025), pp. 877–892, DOI: [10.1007/s11191-024-00530-2](https://doi.org/10.1007/s11191-024-00530-2).
- [11] N. GARDELLA, R. PETTIT, S. L. RIGGS: *Performance, workload, emotion, and self-efficacy of novice programmers using AI code generation*, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education, 2024, pp. 290–296, DOI: [10.1145/3649217.3653615](https://doi.org/10.1145/3649217.3653615).
- [12] S. GROOTHUIJSEN, A. BEEMT, J. C. REMMERS, L. W. MEEUWEN: *AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning*, Computers and Education: Artificial Intelligence 7 (2024), p. 100290, DOI: [10.1016/j.caeai.2024.100290](https://doi.org/10.1016/j.caeai.2024.100290).
- [13] Q. HUANG, C. LV, L. LU, T. SHUANG: *Evaluating the Quality of AI-Generated Digital Educational Resources for University Teaching and Learning*, Systems 13.3 (2025), p. 174, DOI: [10.3390/systems13030174](https://doi.org/10.3390/systems13030174).
- [14] D. M. JOHNSON, W. DOSS, C. M. ESTEPP: *Using ChatGPT with novice Arduino programmers: Effects on performance, interest, self-efficacy, and programming ability*, Journal of Research in Technical Careers 8.1 (2024), DOI: [10.9741/2578-2118.1152](https://doi.org/10.9741/2578-2118.1152).
- [15] M. KAZEMITABAAR, J. CHOW, C. K. T. MA, B. J. ERICSON, D. WEINTROP, T. GROSSMAN: *Studying the effect of AI code generators on supporting novice learners in introductory programming*, in: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, Hamburg, Germany, 2023, DOI: [10.1145/3544548.3580919](https://doi.org/10.1145/3544548.3580919).
- [16] L. LABADZE, M. GRIGOLIA, L. MACHAIDZE: *Role of AI chatbots in education: systematic literature review*, Int J Educ Technol High Educ 20 (2023), p. 56, DOI: [10.1186/s41239-023-00426-1](https://doi.org/10.1186/s41239-023-00426-1).
- [17] M. LEPP, J. KAIMRE: *Does generative AI help in learning programming: Students' perceptions, reported use and relation to performance*, Computers in Human Behavior Reports 18 (2025), p. 100642, DOI: [10.1016/j.chbr.2025.100642](https://doi.org/10.1016/j.chbr.2025.100642).

- [18] Y. LIU, T. LE-CONG, R. WIDYASARI, C. TANTITHAMTHAVORN, L. LI, X. B. D. LE, D. LO: *Refining ChatGPT-Generated Code: Characterizing and Mitigating Code Quality Issues*, ACM Transactions on Software Engineering and Methodology 33.5 (2024), pp. 1–26, DOI: [10.1145/3643674](https://doi.org/10.1145/3643674).
- [19] B. MA, L. CHEN, S. KONOMI: *Enhancing Programming Education with ChatGPT: A Case Study on Student Perceptions and Interactions in a Python Course*, 2024, DOI: [10.48550/arXiv.2403.15472](https://doi.org/10.48550/arXiv.2403.15472), eprint: [arXiv:2403.15472](https://arxiv.org/abs/2403.15472).
- [20] A. PARASURAMAN, C. COLBY: *An Updated and Streamlined Technology Readiness Index: TRI 2.0*, Journal of Service Research 18.1 (2015), DOI: [10.1177/109467051453973](https://doi.org/10.1177/109467051453973).
- [21] A. PARK, T. KIM: *Code suggestions and explanations in programming learning: Use of ChatGPT and performance*, The International Journal of Management Education 23.2 (2025), p. 101119, DOI: [10.1016/j.ijme.2024.101119](https://doi.org/10.1016/j.ijme.2024.101119).
- [22] J. PRATHER, B. N. REEVES, J. LEINONEN, S. MACNEIL, A. S. RANDRIANASOLO, B. A. BECKER, B. KIMMEL, J. WRIGHT, B. BRIGGS: *The widening gap: The benefits and harms of generative AI for novice programmers*, in: Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1, 2024, pp. 469–486, DOI: [10.1145/3632620.3671116](https://doi.org/10.1145/3632620.3671116).
- [23] M. M. RAHMAN, Y. WATANOBE: *ChatGPT for education and research: Opportunities, threats, and strategies*, Applied Sciences 13.9 (2023), p. 5783, DOI: [10.3390/app13095783](https://doi.org/10.3390/app13095783).
- [24] M. M. RAHMAN, Y. WATANOBE, R. U. KIRAN, R. KABIR: *A stacked bidirectional lstm model for classifying source codes built in mpls*, in: Proceedings of the Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021, 2021, pp. 75–89, DOI: [10.1007/978-3-030-93733-1_5](https://doi.org/10.1007/978-3-030-93733-1_5).
- [25] M. M. RAHMAN, Y. WATANOBE, K. NAKAMURA: *A neural network based intelligent support model for program code completion*, Sci. Program. 2020 (2020), p. 7426461, DOI: [10.1155/2020/7426461](https://doi.org/10.1155/2020/7426461).
- [26] M. M. RAHMAN, Y. WATANOBE, K. NAKAMURA: *Source code assessment and classification based on estimated error probability using attentive LSTM language model and its application in programming education*, Appl. Sci. 10 (2020), p. 2973, DOI: [10.3390/app10082973](https://doi.org/10.3390/app10082973).
- [27] P. RAJABI, P. TAGHIPOUR, D. CUKIERMAN, T. DOLECK: *Unleashing ChatGPT's impact in higher education: Student and faculty perspectives*, Computers in Human Behavior: Artificial Humans 2.2 (2024), p. 100090, DOI: [10.1016/j.chbah.2024.100090](https://doi.org/10.1016/j.chbah.2024.100090).
- [28] D. RAVSELI, D. KERŽIČ, N. TOMAŽEVIČ, L. UMEK, N. BREZOVAR, N. A. IAHAD, ET AL.: *Higher education students' perceptions of ChatGPT: A global study of early reactions*, PLoS ONE 20.2 (2025), e0315011, DOI: [10.1371/journal.pone.0315011](https://doi.org/10.1371/journal.pone.0315011).
- [29] M. RICHARDS, K. WAUGH, M. SLAYMAKER, M. PETRE, J. WOODTHORPE, D. GOOCH: *Bob or bot: Exploring ChatGPT's answers to university computer science assessment*, ACM Transactions on Computing Education 24.5 (2024), pp. 1–32, DOI: [10.1145/3633287](https://doi.org/10.1145/3633287).
- [30] A. SARANTI, B. TARAGHI, M. EBNER, A. HOLZINGER: *Property-based testing for parameter learning of probabilistic graphical models*, in: Lncs: 12279. CD-MAKE 2020, ed. by A. HOLZINGER, P. KIESEBERG, A. M. TJOA, E. WEIPPL, 2020, pp. 499–515, DOI: [10.1007/978-3-030-57321-8_28](https://doi.org/10.1007/978-3-030-57321-8_28).
- [31] A. ŠARČEVIĆ, I. TOMIČIĆ, A. MERLIN, M. HORVAT: *Enhancing Programming Education with Open-Source Generative AI Chatbots*, in: 2024 47th MIPRO ICT and Electronics Convention (MIPRO), Opatija, Croatia, 2024, pp. 2051–2056, DOI: [10.1109/MIPRO60963.2024.10569736](https://doi.org/10.1109/MIPRO60963.2024.10569736).
- [32] D. SOBANIA, M. BRIESCH, F. ROTHLAUF: *Choose your programming copilot: A comparison of the program synthesis performance of github copilot and genetic programming*, in: Proceedings of the Genetic and Evolutionary Computation Conference, New York, NY, USA, 2022, pp. 1019–1027, DOI: [10.1145/3512290.3528700](https://doi.org/10.1145/3512290.3528700).

- [33] S. STEINERT, K. E. AVILA, S. RUZIKA, J. KUHN, S. KÜCHEMANN: *Harnessing large language models to develop research-based learning assistants for formative feedback*, Smart Learn. Environ. 11 (2024), p. 62, DOI: [10.1186/s40561-024-00354-1](https://doi.org/10.1186/s40561-024-00354-1).
- [34] D. SUN, A. BOUDOUAIA, C. ZHU, Y. LI: *Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study*, International Journal of Educational Technology in Higher Education 21 (2024), p. 14, DOI: [10.1186/s41239-024-00446-5](https://doi.org/10.1186/s41239-024-00446-5).
- [35] Y. XUE, H. CHEN, G. R. BAI, R. TAIRAS, Y. HUANG: *Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1*, in: ICSE-SEET '24: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, 2024, pp. 331–341, DOI: [10.1145/3639474.3640076](https://doi.org/10.1145/3639474.3640076).
- [36] R. YILMAZ, F. G. K. YILMAZ: *The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation*, Computers & Education: Artificial Intelligence 4 (2023), DOI: [10.1016/j.caeai.2023.100147](https://doi.org/10.1016/j.caeai.2023.100147).
- [37] Q. ZHANG, C. FANG, Y. MA, W. SUN, Z. CHEN: *A Survey of Learning-based Automated Program Repair*, 2023, DOI: [10.1145/3631974](https://doi.org/10.1145/3631974), eprint: [arXiv:2301.03270](https://arxiv.org/abs/2301.03270).

AI-driven fault diagnosis from textual system logs

Áron Kiss, Károly Nehéz, Olivér Hornyák

University of Miskolc
aron.kiss@uni-miskolc.hu
károly.nehez@uni-miskolc.hu
oliver.hornyak@uni-miskolc.hu

Abstract. The increasing complexity and scale of microservice-based systems pose major challenges for ensuring reliability and operational continuity. Multimodal fault diagnosis integrating logs, metrics, and traces has emerged as a key approach for improving anomaly detection, failure type identification, and root cause localization. Graph Neural Networks (GNNs) show strong potential for modeling intricate service dependencies and fault propagation patterns in such systems. This study presents a systematic review of state-of-the-art graph-based multimodal diagnostic frameworks. We compare existing methods in terms of diagnostic accuracy, scalability, computational cost, and implementation complexity, and analyze representative public datasets and benchmark systems. We highlight key challenges, including generalization, explainability, online applicability, and outline promising directions for future research. In addition, we report preliminary findings from our own experiments, which suggest that Transformer-based models provide a promising foundation for multimodal fault diagnosis in enterprise microservice systems. These early results motivate our ongoing work toward hybrid architectures that combine the strengths of Transformers and GNNs.

Keywords: microservice system, fault diagnosis, anomaly detection, root-cause localization, Graph Neural Network, Transformer

AMS Subject Classification: Primary: 68T05, Secondary: 68T30

1. Introduction

The microservice architecture is increasingly prevalent in modern information systems. However, ensuring reliability in such environments – composed of numerous

interdependent services – remains challenging, as faults can propagate in complex ways [13]. To monitor system health, logs, performance metrics, and traces are typically collected. Different anomalies may appear in different modalities: a failure may cause only a metric spike or manifest as log entries indicating a crash [29].

Multimodal fault diagnosis is thus essential and has gained growing attention in recent years [7, 30]. Recent studies focus on combining heterogeneous telemetry data to improve fault detection and root cause analysis.

Graph-based methods are particularly promising [7, 11, 30], since microservices form natural graphs, with nodes as service instances and edges denoting dependencies. Graph Neural Networks (GNNs) leverage this structure to model system-wide interactions [12, 23, 26, 29, 31, 32].

This paper reviews multimodal fault diagnosis methods using GNNs. We compare approaches that integrate log, metric, and trace data across multiple dimensions such as accuracy, scalability, and complexity. We also examine common datasets and benchmarks, and identify research gaps.

In addition, we present preliminary findings from our own work. We first analyze log anomaly detection, showing the benefit of Transformer architectures, then introduce a Graph Transformer-based model for enhanced root cause localization and failure type identification.

2. Methodology

The studies reviewed in this paper were identified using keyword-based searches on platforms of Google Scholar and Semantic Scholar. To ensure comprehensive coverage, we used a diverse set of keywords, including but not limited to: “multimodal fault diagnosis,” “microservice anomaly detection,” “graph neural network,” “data fusion,” “root cause analysis,” and “AIOps benchmark.” The keywords were combined with Boolean operators and adapted iteratively during the search process. Initial selections were filtered based on relevance, citation impact, and publication recency, followed by citation chaining. Although the process was exploratory, we aimed to ensure broad and representative coverage of methodologies and key contributions across both survey and empirical research papers.

3. Multimodal fault diagnosis in microservice systems

Artificial Intelligence for IT Operations (AIOps) aims to enhance IT systems using AI techniques. Despite advancements, challenges in adaptivity, efficiency, and scalability remain, partly due to the lack of standardized taxonomies for data handling, target tasks, and system requirements [13]. This complicates the comparison and integration of diagnostic approaches.

Early fault diagnosis methods in microservice systems typically relied on a single telemetry source, which proved insufficient. Multimodal fault diagnosis addresses

this by jointly analyzing logs, metrics, and traces, capturing complementary system perspectives [12, 29].

For example, a complex outage may manifest through spikes in CPU usage, increased log severity, and timeout traces. Analyzing only one modality could lead to missed or misattributed faults.

The most commonly used data types are:

- **Logs:** Semi-structured textual entries containing system and application events [26]. Log-based methods rely on parsing and pattern mining [20].
- **Metrics:** Time-series data capturing quantitative performance indicators (e.g., CPU usage, request counts) [32]. Diagnostics often involve anomaly detection on time series [20].
- **Traces:** Sequences of spans representing end-to-end request paths. Useful for identifying bottlenecks and fault propagation [30].

Key diagnostic tasks include:

- **Anomaly Detection (AD):** Binary classification to detect abnormal system behavior.
- **Failure Type Identification (FTI):** Multi-class classification to determine the nature of the fault.
- **Root Cause Localization (RCL):** Identifying the component responsible for the fault, often ranked by likelihood; advanced methods also reconstruct propagation paths.

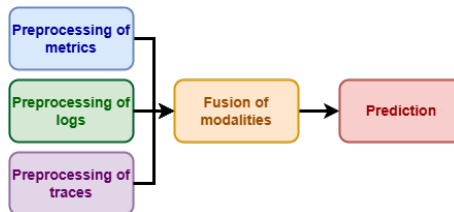


Figure 1. Multimodal fault diagnosis pipeline.

As shown in Figure 1, the diagnosis process typically involves: (a) preprocessing and unifying telemetry data; (b) fusing multimodal features into a model input; and (c) applying predictive models to perform AD, FTI, and RCL.

Modern approaches utilize deep learning to model complex system dynamics. Early works like DeepLog [5] applied sequential modeling to log data. Similarly, LSTM-based models and statistical methods have been used for metric anomaly detection. However, unimodal analysis often lacks sufficient insight, prompting recent methods to integrate all three modalities [26, 29], yielding improved FTI and RCL performance.

4. Graph Neural Networks in fault diagnosis of microservice systems

Graph Neural Networks (GNNs) have become a key trend in fault analysis of microservice systems. These deep learning models are designed to process graph-structured data, making them ideal for modeling microservices, where nodes represent instances and edges denote service dependencies. GNNs can capture both topological and contextual information using techniques like message passing and attention [8, 9, 27].

In multimodal diagnosis, GNNs are used to fuse heterogeneous telemetry data or to process node-level features within a system graph.

Several recent models adopt similar GNN-based architectures [7], differing mainly in the graph encoders and supported diagnostic tasks. Table 1 summarizes their key characteristics.

Table 1. Overview of GNN-based multimodal fault diagnosis models. M, L, T are metrics, logs, and traces, respectively.

Model	Year	Modalities	Graph Encoder	AD	FTI	RCL
Eadro [12]	2023	M,L,T	GAT [2]	✓	✗	✓
DiagFusion [29]	2023	M,L,T	TAGConv [4]	✓	✓	✓
UniDiag [31]	2024	M,L,T	R-GCN [22]	✓	✓	✗
DeepHunt [23]	2024	M,L,T	GraphSAGE [8]	✓	✗	✓
TVDiag [26]	2025	M,L,T	GraphSAGE [8]	✓	✓	✓
CHASE [32]	2025	M,L,T	HGT [9]	✓	✗	✓

Eadro [12] uses GAT over fused multimodal inputs, but its complexity may cause overfitting on small systems, where simpler models sometimes perform better [7].

DiagFusion [29] embeds alerts using fastText and fuses features via a multi-part neural architecture. However, ignoring modality-specific traits may harm performance when data is incomplete [24].

UniDiag [31] builds temporal knowledge graphs and uses R-GCN for efficient stream-based diagnosis, introducing a microservice-specific embedding strategy.

TVDiag [26] avoids uniform fusion by using task-specific GNN encoders and contrastive learning, improving robustness via graph augmentation.

DeepHunt [23] introduces Root Cause Score (RCS), combining reconstruction error and propagation patterns. It also supports adaptive refinement via feedback.

CHASE [32] employs a causal hypergraph framework with modality weighting and inference to identify root causes. While accurate, its resource demands may limit scalability.

In summary, GNN-based models effectively integrate multimodal data and capture complex patterns. Still, simpler models with well-designed features can match

or exceed GNN performance in constrained settings [7].

5. Public datasets and benchmark environments

A major challenge in fault diagnosis research is the scarcity of large-scale, publicly available datasets from real-world microservice systems [13]. Consequently, many studies rely on proprietary data or synthetic environments. In recent years, several datasets have been released to support model development and evaluation.

Table 2 summarizes widely used datasets for FTI and RCL tasks, including both single and multimodal sources.

Table 2. Overview of widely used, openly available information system anomaly datasets. M, L, and T are metrics, logs, and traces, respectively.

Name	Year	Modalities	Description
AIOps Challenge 2020 [16]	2020	M, T	Data about a real-world production microservice system, collected by China Mobile Zhejiang.
AIOps Challenge 2021 [1]	2021	M, L, T	Data about two large commercial banking systems, collected by Tsinghua University.
Generic AIOps Atlas (GAIA) [3]	2022	M, L, T	Detailed dataset with simulated anomalies and injection records for root cause analysis evaluation, collected by CloudWise.
Loghub [33]	2023	L	Large volume of logs from various systems.
RCAEval-RE1 [17, 18]	2024	M	Metric-only failure cases from three microservice systems, supporting metric-based root cause analysis.
RCAEval-RE2, RCAEval-RE3 [18]	2025	M, L, T	Multi-source telemetry data for root cause analysis, with RE2 covering general failure cases and RE3 focusing on code-level fault cases across three microservice systems.

These resources facilitate benchmarking and comparison of diagnostic models. However, most datasets involve synthetic fault injection, which may fail to capture the complexity and spontaneity of real production failures. As a result, models trained exclusively on these may face generalization issues in large-scale industrial settings.

In addition to datasets, open-source benchmark systems have been developed to emulate microservice environments with integrated telemetry tools (e.g., OpenTelemetry, Prometheus, Jaeger, Loki). Table 3 lists commonly used examples.

Several studies inject faults into these systems and monitor the telemetry to assess model performance [12, 26]. While effective for controlled experimentation, their scale – typically 8–12 services – is far from enterprise-grade systems. For context, Netflix operated over 200 microservices by 2016 [6], while Uber reported 4,500 by 2023 [25].

Table 3. Overview of widely used microservice benchmark systems.

Name	Release	Microservices	URL
SockShop	2016	8	https://github.com/microservices-demo/microservices-demo
Train Ticket	2018	41	https://github.com/FudanSELab/train-ticket
Hotel Reservation	2019	10	https://github.com/delimitrou/DeathStarBench/tree/master/hotelReservation
Social Network	2019	12	https://github.com/delimitrou/DeathStarBench/tree/master/socialNetwork
Online Boutique	2020	12	https://github.com/GoogleCloudPlatform/microservices-demo

6. Comparison of existing methods

Evaluating multimodal fault diagnosis methods requires careful consideration of several criteria. In this section, we summarize the most important dimensions for comparison, with a particular focus on detection accuracy, computational cost and scalability, real-time applicability, and implementation complexity. We also discuss how recent models perform with respect to each of these aspects.

6.1. Detection and localization accuracy

In general, multimodal GNN-based models tend to achieve the improved reported accuracy, compared to single modal approaches in published studies. A recent evaluation [7] showed that state-of-the-art models such as Eadro, DiagFusion, DeepHunt, TVDiag, and CHASE achieved top-1 hit rates between 30% and 60% on the GAIA dataset, while their top-3 hit rates ranged from approximately 59% to 88%. Precision is another critical metric, particularly when assessing false positives. Multimodal models tend to outperform single-modality detectors in this regard, as they require consistent signals across multiple modalities to trigger an alert – resulting in fewer spurious detections.

However, it is important to note that the advantage of multimodal GNNs in accuracy may be highly dependent on the dataset and fault types. Some studies suggest that in certain scenarios, this advantage may not be statistically significant [7].

6.2. Computational cost

Graph-based deep learning models are generally more computationally expensive than simpler architectures such as MLPs or LSTMs. This is due to the complexity of GNN layers (e.g., graph convolution or attention operations that involve neighborhood aggregation with time complexity $O(V + E)$ depending on the graph structure), and the additional overhead introduced by multimodal data preprocessing (e.g., log parsing, metric filtering, dimensionality reduction). A key scalability concern is that most GNN-based diagnostic models have not been tested in online setups, and on systems with hundreds of service instances.

Although current datasets are manageable with GNNs, it remains unclear whether these methods can scale effectively to larger environments. Future progress may require more scalable architectures, as the computational cost of running complex multimodal models may be prohibitive in smaller organizational settings lacking adequate infrastructure.

6.3. Real-time applicability

A practical requirement for fault diagnosis methods is their ability to operate in real time or near real time. Most published multimodal models are trained offline and applied in a near-real-time setting, where the model processes telemetry aggregated over a fixed interval (e.g., every second or minute) and raises alerts as needed. Offline training is not inherently problematic if model updates are infrequent. However, microservice environments are highly dynamic, raising the need for continual learning or online inference capabilities. Real-time applicability is further constrained by the latency of preprocessing steps such as log parsing, metric filtering, and trace clustering. Stream-based tools like *Drain3* for logs and sliding-window aggregation for metrics are essential in online systems.

There is broad consensus in the literature that current deep learning models require further optimization, such as model pruning or knowledge distillation to be truly viable in real-time AIOps settings. For now, multimodal GNN-based models are best suited for near real time detection.

6.4. Engineering complexity

Implementing multimodal GNN-based diagnostic systems demands significant engineering effort. Key challenges include handling and synchronizing three different telemetry streams (logs, metrics, and traces), cleaning and storing diverse data sources, and integrating multiple specialized algorithms (e.g. NLP for log analysis, time-series modeling for metrics, and graph processing for traces). In many implementations, the diagnostic pipeline comprises multiple interdependent components that must be individually tuned [26, 29]. For instance, DiagFusion and TVDiag include a full preprocessing framework before the neural network model begins learning. This involves filtering logs, normalizing metrics, and deriving service dependency graphs from trace data.

GNN models typically require dedicated libraries such as *PyTorch Geometric* or *Deep Graph Library*, which introduce a steeper learning curve for operations teams. Moreover, debugging end-to-end pipelines can be difficult: when the diagnosis fails, it is not always clear whether the error occurred during log parsing, feature fusion, or within the GNN itself. This may prevent the utilization of such methods for small-medium sized or non-technology-focused organizations with less AI expertise.

In conclusion, there is no single best method that dominates across all dimensions. GNN-based deep learning models offer high accuracy and effectively leverage multimodal data. However, they come with considerable resource demands, scalability limitations, and operation complexity.

7. Limitations and open challenges

Despite recent progress in multimodal fault diagnosis for microservice systems, several important challenges and limitations remain. These arise both from the limitations of current methods and the shortcomings of existing datasets and benchmarks. Below, we highlight the most critical issues identified.

7.1. Generalizability

Most published models have been evaluated on relatively small and simple fault scenarios. It is common for studies to assess their methods on 1-2 microservice benchmarks with just 2-5 injected fault types [12, 29, 31]. As a result, the generalizability of these models to real-world systems remains unclear. In production environments, dozens of distinct root causes may exist, including hardware failures, code-level bugs, and network anomalies, many of which are not represented in the training data. Moreover, operational environments are constantly evolving

due to factors of sudden peaks in user demand, ongoing software development, and the introduction of new hardware components, further complicating fault diagnosis and model robustness.

For example, a GNN model tuned primarily for detecting CPU spikes may completely miss a database deadlock scenario if neither the metrics nor the log anomaly detection modules provide a clear signal. Overfitting to the limited fault cases present in the training set is a common risk – as demonstrated by Eadro, which performed well on trained scenarios but underperformed in unseen environments [7]. Current deep learning models lack proven generalization capabilities, and there is a strong need for larger, more diverse benchmarks that reflect real production complexity.

7.2. Lack of real-world data

Although public datasets exist, many studies still rely on synthetic fault injection. While such datasets can approximate real-world conditions, they are inherently artificial. In actual operations, failures often arise from overlapping and interdependent causes, such as a minor memory leak combined with a configuration change, which are difficult to reproduce under controlled lab conditions. This limits the external validity of many evaluation results.

A major reason for the lack of real-world data is the sensitivity of production environments. Sharing logs and failure cases may pose risks related to business confidentiality, intellectual property, or security. However, it is also in the interest of organizations that researchers develop methods targeting real operational challenges. Without access to realistic data, proposed models may remain overfitted to synthetic scenarios and fail to generalize to production-scale systems.

Furthermore, synthetic datasets typically include ground truth labels for the faulty component, allowing models to be trained to recognize known faults rather than to discover novel root causes in unfamiliar environments. This introduces potential bias. To truly evaluate and improve fault diagnosis systems, publicly available datasets based on real-world incidents and complex causal chains are needed.

7.3. Interpretability and explainability

Deep learning models, particularly GNNs, often lack transparent decision-making mechanisms. In operational contexts, it is critical for human operators to understand why a service was flagged as faulty. Although recent studies have begun to address this issue, most state-of-the-art models still fall short in terms of explainability.

For instance, a GNN might output *Node X is anomalous (confidence = 0.9)*, but this information is of limited practical use, unless the system can also indicate which features contributed to this decision. Addressing this challenge requires either post hoc interpretation techniques (e.g., feature attribution or counterfactual analysis), or the integration of inherently interpretable components into the

model architecture. While attention mechanisms have been used in some models to provide a degree of transparency, their alignment with human-understandable explanations remains limited and context-dependent. Methods such as SHAP [15], LIME [21], and GNNExplainer [28] have been proposed to improve model interpretability by identifying the most influential features or substructures contributing to a prediction.

7.4. Scalability

Most current methods have not been tested on industrial-scale systems with at least 50–100 microservices. GNN complexity typically grows linearly with the number of nodes and edges, but can become quadratic with operations such as self-attention over large node sets. In large-scale environments, the volume of telemetry data and the complexity of service interactions may significantly impact inference time and memory usage, raising concerns about real-time applicability and deployment feasibility.

Furthermore, large microservice systems are likely to experience multiple concurrent faults, which may be independent or interacting. Supporting multi-fault scenarios increases the diagnostic complexity of models, as they must disambiguate overlapping symptoms and attribute them to distinct root causes. This additional burden further challenges the scalability of current approaches and highlights the need for more robust, efficient, and fault-tolerant diagnostic architectures.

7.5. Preprocessing and pipeline fragility

Multimodal diagnostic pipelines depend on complex, often computationally intensive preprocessing steps, such as log parsing, metric normalization, trace correlation, and graph construction. These can strongly influence downstream model performance. The diversity of current approaches has led to heterogeneous workflows, complicating reproducibility and fair comparison.

Moreover, errors or inconsistencies during preprocessing can severely degrade diagnostic accuracy. For instance, log parsing failures may cause misclassification, while incomplete traces from sampling or missing instrumentation lead to fragmented service graphs. Noisy or outlier-heavy metric series can distort model behavior and degrade learned representations.

Current models often lack robustness to such imperfections. Beyond improving resilience via adaptive preprocessing or uncertainty-aware learning, there is a clear need for standardization. Common data formats and pipelines would support more meaningful comparisons and enable sharing of preprocessed datasets alongside raw telemetry. This would lower the entry barrier for new researchers and promote transparency and reproducibility.

8. Preliminary explorations to address challenges

To assess how different AI models perform in detecting faults based on event sequences, we conducted an initial comparative evaluation of machine learning approaches for log anomaly detection. Our focus was on event sequence-based methods that transform logs into sequences of event template identifiers and classify them as either normal or anomalous. This process is depicted in Figure 2.

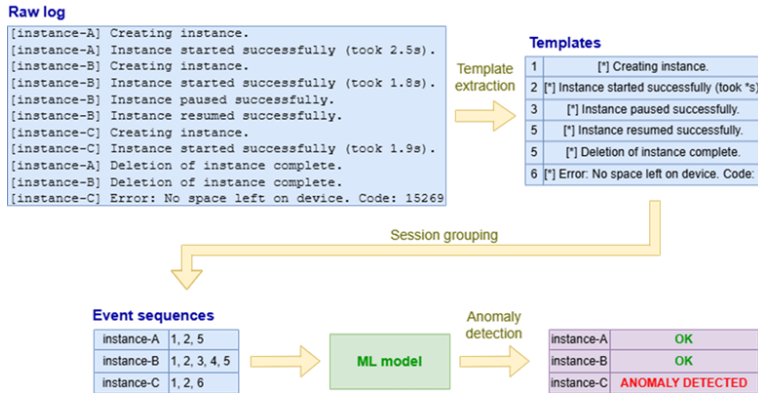


Figure 2. Process of event sequence based log anomaly detection.

Using the HDFS dataset [33], we benchmarked four models: the original LSTM-based DeepLog implementation [5], as well as our own variations based on a bidirectional LSTM, a Random Forest classifier, and a Transformer encoder.

In our experiments, the Transformer model outperformed the LSTM baseline, Bi-LSTM, and Random Forest in terms of overall accuracy, precision, and F1-score. Its ability to capture long-range dependencies resulted in $\sim 6\%$ gain in F1-score over the LSTM and a $\sim 25\%$ gain over the Random Forest. Notably, it also achieved about 11% relative improvement over the baseline LSTM model in precision. This indicates the model’s superior ability to reduce false positives while maintaining high recall. These findings suggest that Transformer-based architectures are a promising choice for anomaly detection, as they effectively capture dependencies in event sequences. Motivated by these results, we began extending this approach to a multimodal diagnostic setting by integrating logs, metrics, and traces into a unified model.

As a first step toward this goal, we developed a prototype system that leverages Transformer-based graph encoders to integrate logs, metrics, and traces for fault diagnosis in microservice systems. Our design choices were informed by key limitations identified in the literature, including the lack of robust multimodal fusion strategies, the limited scalability of conventional GNNs in large microservice architectures, and the absence of interpretability mechanisms suitable for operational use.

Our method builds on the TVDiag framework [26], replacing its GraphSAGE-based encoder with the Graphormer architecture [27], a Transformer model adapted for graphs. As illustrated in Figure 3, telemetry anomalies are detected using tailored strategies: the 3σ method [19] for metrics, Isolation Forest [14] for traces, and pattern-based filtering for logs. Alerts are embedded using fastText [10] and mapped to nodes in a graph representing service instances. Edges are formed based on runtime communication patterns between the instances.

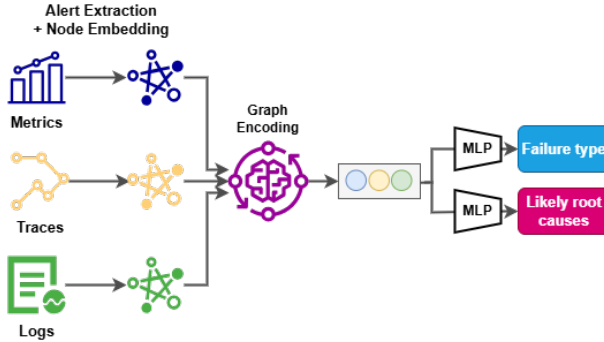


Figure 3. Overview of our prototype framework for multimodal fault diagnosis.

To overcome the lack of inherent node ordering in graphs, we adopted encoding strategies introduced by [27]. Node centrality – computed from node degrees – is used to highlight structurally important service instances, which are more likely to play critical role in failures. Additionally, spatial encoding captures shortest-path distances between nodes, allowing the model to account for fault propagation patterns within the service graph. Together, these encodings offer a richer structural context that can support more informed fault reasoning.

The encoded graph is passed through Graphormer layers, followed by two MLP heads performing FTI and RCL. We performed preliminary evaluation using the Generic AIOps Atlas dataset, which features fault scenarios in a QR-based access control system. Compared to the baseline TVDiag model, our method yielded 3–6% higher RCL top-1 to top-3 hit rate and approximately 5% improvement in F1-score for FTI, while reducing inference time by up to 80%, thanks to the model’s parallelizable architecture.

Although further validation is needed, results show that combining Transformer-based graph representation learning with multimodal data can address certain key challenges in fault diagnosis. In particular, it improves both diagnostic accuracy and runtime efficiency, while providing interpretable features like centrality and spatial locality that can assist system administrators. Our work is still in an early stage and should not be interpreted as a complete solution. Instead, it offers encouraging evidence to justify deeper investigation. Future work will focus on deploying the model in dynamic, real-time settings and addressing issues such as data imbal-

ance and preprocessing overhead. Understanding how the model performs under online conditions and dynamically evolving service topologies will be essential for evaluating its practical value in production systems.

9. Conclusion

In recent years, multimodal fault diagnosis in microservice systems has made significant progress, especially in combining logs, metrics, and traces to improve fault detection and root cause localization. GNNs have shown strong potential for modeling fault propagation across complex service topologies, offering notable gains in detection accuracy and root cause localization.

Despite these benefits, GNNs demand significant computational resources and are highly sensitive to data quality. Interestingly, simpler models, such as multilayer perceptrons can also be competitive when paired with well-designed preprocessing and feature engineering.

Current research indicates that no single model consistently outperforms others across all scenarios. Choosing the right approach involves trade-offs between accuracy, scalability, automation, and interpretability. In industrial settings, GNN-based models are still rarely deployed for real-time fault detection. A promising strategy is to use hybrid architectures: lightweight models for routine monitoring, complemented by GNN-based diagnostics for critical or ambiguous cases.

Achieving practical breakthroughs will require greater focus on addressing real-world constraints, such as data availability, benchmark realism, and model transparency, rather than solely increasing model complexity. Our ongoing research, starting from single-modal Transformer-based log anomaly detection and advancing toward a graph-based multimodal framework, provides early evidence that appropriately adapted Transformer models may offer substantial benefits. However, thorough experimentation and validation remain necessary to evaluate the robustness and generalizability of this approach.

References

- [1] AIOps-NANKAI: *AIOps2021*, 2021, URL: <https://www.aiops.cn/gitlab/aiops-nankai/data/trace/aiops2021/-/tree/main> (visited on 06/15/2025).
- [2] S. BRODY, U. ALON, E. YAHAV: *How Attentive are Graph Attention Networks?*, 2022, arXiv: 2105.14491 [cs.LG], URL: <https://arxiv.org/abs/2105.14491>.
- [3] CLOUDWISE: *Generic AIOps Atlas*, 2022, URL: <https://github.com/CloudWise-OpenSource/GAIA-DataSet> (visited on 06/15/2025).
- [4] J. DU, S. ZHANG, G. WU, J. M. F. MOURA, S. KAR: *Topology Adaptive Graph Convolutional Networks*, 2018, arXiv: 1710.10370 [cs.LG], URL: <https://arxiv.org/abs/1710.10370>.
- [5] M. DU, F. LI, G. ZHENG, V. SRIKUMAR: *DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning*, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, ACM, Oct. 2017, pp. 1285–1298, DOI: 10.1145/3133956.3134015, URL: <http://dx.doi.org/10.1145/3133956.3134015>.

- [6] Y. GAN, Y. ZHANG, D. CHENG, A. SHETTY, P. RATHI, N. KATARKI, A. BRUNO, J. HU, B. RITCHKEN, B. JACKSON, K. HU, M. PANCHOLI, Y. HE, B. CLANCY, C. COLEN, F. WEN, C. LEUNG, S. WANG, L. ZARUVINSKY, M. ESPINOSA, R. LIN, Z. LIU, J. PADILLA, C. DELIMITROU: *An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud - Edge Systems*, in: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '19, ACM, Apr. 2019, pp. 3–18, DOI: [10.1145/3297858.3304013](https://doi.org/10.1145/3297858.3304013), URL: <http://dx.doi.org/10.1145/3297858.3304013>.
- [7] F. GAO, R. XIN, X. LI, Y. ZHANG: *Are GNNs Actually Effective for Multimodal Fault Diagnosis in Microservice Systems?*, 2025, DOI: [10.48550/ARXIV.2501.02766](https://doi.org/10.48550/ARXIV.2501.02766), URL: <https://arxiv.org/abs/2501.02766>.
- [8] W. L. HAMILTON, R. YING, J. LESKOVEC: *Inductive Representation Learning on Large Graphs*, 2018, arXiv: [1706.02216](https://arxiv.org/abs/1706.02216) [cs.SI], URL: <https://arxiv.org/abs/1706.02216>.
- [9] Z. HU, Y. DONG, K. WANG, Y. SUN: *Heterogeneous Graph Transformer*, in: Proceedings of The Web Conference 2020, WWW '20, ACM, Apr. 2020, pp. 2704–2710, DOI: [10.1145/3366423.3380027](https://doi.org/10.1145/3366423.3380027), URL: <http://dx.doi.org/10.1145/3366423.3380027>.
- [10] A. JOULIN, E. GRAVE, P. BOJANOWSKI, T. MIKOLOV: *Bag of Tricks for Efficient Text Classification*, 2016, DOI: [10.48550/ARXIV.1607.01759](https://doi.org/10.48550/ARXIV.1607.01759), URL: <https://arxiv.org/abs/1607.01759>.
- [11] L. KOVÁCS, E. BAKSÁNE VARGA, P. MILEFF: *Prediction of Complex Event Graphs with Neural Networks*, Computing and Informatics 43.1 (2024), pp. 181–212, ISSN: 2585-8807, DOI: [10.31577/cai_2024_1_181](https://doi.org/10.31577/cai_2024_1_181), URL: http://dx.doi.org/10.31577/cai_2024_1_181.
- [12] C. LEE, T. YANG, Z. CHEN, Y. SU, M. R. LYU: *Eadro: An End-to-End Troubleshooting Framework for Microservices on Multi-source Data*, 2023, DOI: [10.48550/ARXIV.2302.05092](https://doi.org/10.48550/ARXIV.2302.05092), URL: <https://arxiv.org/abs/2302.05092>.
- [13] Z. LI, N. ZHAO, S. ZHANG, Y. SUN, P. CHEN, X. WEN, M. MA, D. PEI: *Constructing Large-Scale Real-World Benchmark Datasets for AIOps*, 2022, DOI: [10.48550/ARXIV.2208.03938](https://doi.org/10.48550/ARXIV.2208.03938), URL: <https://arxiv.org/abs/2208.03938>.
- [14] F. T. LIU, K. M. TING, Z.-H. ZHOU: *Isolation Forest*, in: 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 413–422, DOI: [10.1109/ICDM.2008.17](https://doi.org/10.1109/ICDM.2008.17).
- [15] S. M. LUNDBERG, S.-I. LEE: *A Unified Approach to Interpreting Model Predictions*, in: Advances in Neural Information Processing Systems 30, ed. by I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN, R. GARNETT, Curran Associates, Inc., 2017, pp. 4765–4774, URL: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>.
- [16] NETMANAIOps: *AIOps-Challenge-2020-Data*, 2020, URL: <https://github.com/NetManAIOps/AIOps-Challenge-2020-Data> (visited on 06/15/2025).
- [17] L. PHAM, H. HA, H. ZHANG: *Root Cause Analysis for Microservice System based on Causal Inference: How Far Are We?*, in: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE '24, ACM, Oct. 2024, pp. 706–715, DOI: [10.1145/3691620.3695065](https://doi.org/10.1145/3691620.3695065), URL: <http://dx.doi.org/10.1145/3691620.3695065>.
- [18] L. PHAM, H. ZHANG, H. HA, F. SALIM, X. ZHANG: *RCAEval: A Benchmark for Root Cause Analysis of Microservice Systems with Telemetry Data*, in: Companion Proceedings of the ACM on Web Conference 2025, WWW '25, ACM, May 2025, pp. 777–780, DOI: [10.1145/3701716.3715290](https://doi.org/10.1145/3701716.3715290), URL: <http://dx.doi.org/10.1145/3701716.3715290>.
- [19] F. PUKELSHEIM: *The three sigma rule*, The American Statistician 48.2 (1994), pp. 88–91, URL: <https://d-nb.info/1186632208/34>.
- [20] Y. REMIL, A. BENDIMERAD, R. MATHONAT, M. KAYTOUE: *AIOps Solutions for Incident Management: Technical Guidelines and A Comprehensive Literature Review*, 2024, arXiv: [2404.01363](https://arxiv.org/abs/2404.01363) [cs.OS], URL: <https://arxiv.org/abs/2404.01363>.

- [21] M. T. RIBEIRO, S. SINGH, C. GUESTRIN: “Why Should I Trust You?”: Explaining the Predictions of Any Classifier, 2016, arXiv: [1602.04938 \[cs.LG\]](#), URL: <https://arxiv.org/abs/1602.04938>.
- [22] M. SCHLICHTKRULL, T. N. KIPF, P. BLOEM, R. VAN DEN BERG, I. TITOV, M. WELLING: *Modeling Relational Data with Graph Convolutional Networks*, in: The Semantic Web, ed. by A. GANGEMI, R. NAVIGLI, M.-E. VIDAL, P. HITZLER, R. TRONCY, L. HOLLINK, A. TORDAI, M. ALAM, Cham: Springer International Publishing, 2018, pp. 593–607, ISBN: 978-3-319-93417-4.
- [23] Y. SUN, Z. LIN, B. SHI, S. ZHANG, S. MA, P. JIN, Z. ZHONG, L. PAN, Y. GUO, D. PEI: *Interpretable Failure Localization for Microservice Systems Based on Graph Autoencoder*, ACM Transactions on Software Engineering and Methodology 34.2 (Jan. 2025), pp. 1–28, ISSN: 1557-7392, DOI: [10.1145/3695999](#), URL: <http://dx.doi.org/10.1145/3695999>.
- [24] L. TAO, S. ZHANG, Z. JIA, J. SUN, M. MA, Z. LI, Y. SUN, C. YANG, Y. ZHANG, D. PEI: *Giving Every Modality a Voice in Microservice Failure Diagnosis via Multimodal Adaptive Optimization*, in: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE ’24, ACM, Oct. 2024, pp. 1107–1119, DOI: [10.1145/3691620.3695489](#), URL: <http://dx.doi.org/10.1145/3691620.3695489>.
- [25] UBER ENGINEERING: *Up: Portable Microservices Ready for the Cloud*, Uber blog post, Sept. 2023, URL: <https://www.uber.com/en-HU/blog/up-portable-microservices-ready-for-the-cloud/> (visited on 06/16/2025).
- [26] S. XIE, J. WANG, H. HE, Z. WANG, Y. ZHAO, N. ZHANG, B. LI: *TVDiag: A Task-oriented and View-invariant Failure Diagnosis Framework with Multimodal Data*, 2024, arXiv: [2407.19711v1 \[cs.SE\]](#), URL: <https://arxiv.org/abs/2407.19711v1>.
- [27] C. YING, T. CAI, S. LUO, S. ZHENG, G. KE, D. HE, Y. SHEN, T.-Y. LIU: *Do Transformers Really Perform Bad for Graph Representation?*, 2021, arXiv: [2106.05234 \[cs.LG\]](#), URL: <https://arxiv.org/abs/2106.05234>.
- [28] R. YING, D. BOURGEOIS, J. YOU, M. ZITNIK, J. LESKOVEC: *GNNEExplainer: Generating Explanations for Graph Neural Networks*, 2019, arXiv: [1903.03894 \[cs.LG\]](#), URL: <https://arxiv.org/abs/1903.03894>.
- [29] S. ZHANG, P. JIN, Z. LIN, Y. SUN, B. ZHANG, S. XIA, Z. LI, Z. ZHONG, M. MA, W. JIN, D. ZHANG, Z. ZHU, D. PEI: *Robust Failure Diagnosis of Microservice System through Multimodal Data*, 2023, DOI: [10.48550/ARXIV.2302.10512](#), URL: <https://arxiv.org/abs/2302.10512>.
- [30] S. ZHANG, S. XIA, W. FAN, B. SHI, X. XIONG, Z. ZHONG, M. MA, Y. SUN, D. PEI: *Failure Diagnosis in Microservice Systems: A Comprehensive Survey and Analysis*, 2024, DOI: [10.48550/ARXIV.2407.01710](#), URL: <https://arxiv.org/abs/2407.01710>.
- [31] S. ZHANG, Y. ZHAO, S. XIA, S. WEI, Y. SUN, C. ZHAO, S. MA, J. KUANG, B. ZHU, L. PAN, Y. GUO, D. PEI: *No More Data Silos: Unified Microservice Failure Diagnosis With Temporal Knowledge Graph*, IEEE Transactions on Services Computing 17.6 (Nov. 2024), pp. 4013–4026, ISSN: 2372-0204, DOI: [10.1109/tsc.2024.3489444](#), URL: <http://dx.doi.org/10.1109/TSC.2024.3489444>.
- [32] Z. ZHAO, Z. WANG, T. ZHANG, Z. SHEN, H. DONG, Z. LEI, X. MA, G. XU, Z. DING, Y. YANG: *CHASE: A Causal Hypergraph based Framework for Root Cause Analysis in Multimodal Microservice Systems*, 2025, arXiv: [2406.19711 \[cs.LG\]](#), URL: <https://arxiv.org/abs/2406.19711>.
- [33] J. ZHU, S. HE, P. HE, J. LIU, M. R. LYU: *Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics*, in: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), IEEE, Oct. 2023, pp. 355–366, DOI: [10.1109/issre59848.2023.00071](#), URL: <http://dx.doi.org/10.1109/ISSRE59848.2023.00071>.

Soft voting robustness in neural network ensembles with empirical analysis and formal verification

Ádám Kovács^{a,b}, Roland Gunics^a,
Gergely Kovásznai^a, Tibor Tajti^{a,b}

^aEszterházy Károly Catholic University
kovacs2.adam@uni-eszterhazy.hu
gunics.roland@uni-eszterhazy.hu
kovasznai.gergely@uni-eszterhazy.hu
tajti.tibor@uni-eszterhazy.hu

^bUniversity of Debrecen, Doctoral School of Informatics

Abstract. Neural network ensembles with soft voting improve accuracy and stability by aggregating multiple models; however, their reliability under individual model failure remains a critical concern. This paper addresses the robustness of soft-voting ensembles in safety-critical settings by combining empirical analysis and formal verification. We evaluate the impact of single-model failures on ensemble performance and find that soft voting yields graceful degradation, with only minimal loss in accuracy when one component model is removed or corrupted. In parallel, we develop a formal verification framework to investigate whether the ensemble’s final prediction remains unchanged under any single-model failure scenario. The results demonstrate that soft-voting ensembles can maintain reliable outputs despite individual model failures, providing both empirical evidence and provable guarantees of fault tolerance in neural network ensembles.

Keywords: neural network, ensemble, robustness, model failure, formal verification, SMT

AMS Subject Classification: 68T07, 68T27, 68T37

1. Introduction

Modern safety-critical applications of AI, such as autonomous driving and health-care, demand not only high accuracy but also *formal robustness guarantees* for

reliability under all conditions. Ensemble learning is a proven approach to improve reliability: by aggregating multiple neural network models, ensembles achieve higher accuracy and greater stability than individual networks. However, empirical robustness (e.g., against noisy or adversarial inputs) achieved via techniques like adversarial training or data augmentation does not automatically translate into formal guarantees of correctness.

In high-stakes domains, we require provable assurance that the system will maintain correct operation even when some components fail or behave unexpectedly. This work introduces the concept of *voting robustness* as a measure of an ensemble’s tolerance to individual model failures, and provides both empirical analysis and formally verified guarantees for this property.

We present a comprehensive study of voting robustness in neural network ensembles, evaluating their resilience across diverse architectures on a digit classification task. Our experiments demonstrate that even if one model in the ensemble is corrupted or fails entirely, the aggregated prediction remains essentially unchanged, with minimal loss in accuracy.

Beyond empirical evaluation, we provide formal verification of these ensembles using neural network verification tools based on Satisfiability Modulo Theories (SMT). By encoding the soft-voting mechanism into a single verifiable model, we can prove if the ensemble’s prediction remains stable under single-model failures for any set of inputs. This dual approach of empirical analysis and formal verification establishes ensemble voting as a powerful mechanism for achieving built-in robustness, paving the way for AI systems that can be deployed with certified reliability in safety-critical domains.

2. Background and related work

2.1. Ensemble learning and voting schemes

Ensemble methods combine multiple learners to improve generalization performance. Classic ensemble techniques such as bagging, boosting, and random forests leverage model diversity to reduce variance and increase accuracy. In neural network ensembles, each model (potentially differing in initialization, architecture, or training data) contributes to the final prediction.

Voting can be performed in two primary ways: *majority voting* (hard voting), where the class with the most votes is selected, and *averaging* (soft voting), where the models’ output probabilities are averaged and the class with the highest mean probability is chosen. Soft voting typically provides smoother decision boundaries and is easier to integrate into verification pipelines, as the voting layer is differentiable.

Prior work has demonstrated that ensembles can enhance adversarial robustness, as diversity among models makes it more difficult for a single adversarial example to mislead the entire ensemble. For instance, it was demonstrated that promoting diversity in non-maximal predictions via an adaptive regularizer en-

hanced both ensemble robustness and transfer resilience [10]. While substantial work has focused on empirical defenses such as adversarial training, noise resilience, and robustness to distributional shifts, much of this remains heuristic and lacks formal guarantees [6, 12]. Gross et al. showed that ensemble robustness verification is NP-hard and proposed SMT- and MILP-based encodings to either find optimal randomized attacks or formally prove robustness [5].

2.2. Voting robustness

We define *voting robustness* as the minimum number of model predictions that must change to alter the ensemble’s final decision. This concept is analogous to the vote margin in classical voting theory.

For a majority vote ensemble of $n > 0$ models, voting robustness is simply the number of votes by which the leading class exceeds the runner-up (e.g., if 6 out of 10 models vote for the predicted class, at least two votes must flip to change the outcome). In soft-voting ensembles, robustness relates to the *confidence margin* – the gap between the averaged probability assigned to the predicted class and the next highest class. A large margin indicates that more substantial changes in individual model outputs are needed to alter the final prediction.

Voting robustness is particularly crucial for safety-critical AI systems, as high robustness ensures that even if some ensemble members fail or behave incorrectly, the system’s overall decision remains unaffected.

2.3. Robustness metrics

To capture an ensemble’s tolerance to component failures, we summarize and define several complementary metrics beyond raw accuracy.

Ensemble Accuracy: The accuracy of the ensemble on a test set under normal conditions, serving as the baseline for comparison.

Accuracy Drop: The reduction in accuracy when one or more models are corrupted or removed. A robust ensemble should degrade gracefully, typically losing less than 1% accuracy when a single member is compromised.

Class Switching Probability (CSP): The probability that the ensemble’s predicted class changes when a single model’s prediction changes. Low CSP values mean that no single model has undue influence on the decision.

Ensemble Margin: The difference between the ensemble’s support for the predicted class and the runner-up class. A larger margin indicates greater decision stability, meaning that individual model failures are less likely to overturn the ensemble’s output.

Leave-One-Out (LOO) Impact: A diagnostic analysis where each model is removed in turn to measure the effect on predictions and accuracy. Significant accuracy drops or frequent class changes upon removal reveal that a model is critical to the ensemble’s decision-making.

2.4. Formal verification of neural networks

The formal methods community has developed several approaches for verifying properties of neural networks, including Satisfiability Modulo Theories (SMT) solving, linear programming (LP), and abstract interpretation. These methods have been applied to tasks such as verifying robustness to input perturbations and ensuring other safety properties [8, 11].

Tools such as Marabou [8, 13] encode a network’s ReLU activation constraints as LP constraints, enabling solvers to determine whether specific outputs can be altered under given conditions. These techniques provide *certified guarantees* (e.g., proving that no adversarial example exists within a bounded input region).

However, most verification research to date has focused on single networks [2, 3, 7]. Ensemble models introduce additional complexity, particularly when using discrete majority voting, which creates combinatorial branching. Encoding majority voting in a formal verification setting is challenging due to the large number of possible vote distributions, significantly increasing the combinatorial complexity.

By contrast, averaging (soft voting) produces continuous outputs that can be expressed as an additional network layer, effectively *fusing* the ensemble into a single verifiable network. This transformation enables existing verification pipelines – whether based on SMT, LP, or abstract interpretation – to be applied to ensemble models without incurring the combinatorial explosion caused by discrete voting schemes.

2.5. Related work

Adversarial training with generated examples has been explored to close the robustness gap, and ensemble adversarial training has demonstrated that ensemble diversity improves defense effectiveness [4].

Formal verification of neural networks is an active field, with recent advances extending verification techniques to a wider class of activation functions and input sets. For example, Antal et al. [1] generalize verification methods for piece-wise linear activation functions, supporting both bounded and unbounded input domains, and demonstrate their effectiveness on multiple case studies.

Our work builds directly on these insights by adopting soft voting for ensemble verification, thereby enabling formal analysis while preserving robustness. To our knowledge, this is among the first studies to *verify ensemble voting robustness formally*, combining empirical results with SMT-based verification to certify that ensemble predictions remain stable even when some members are corrupted or removed.

3. Methodology

We designed an experiment to empirically evaluate the robustness of ensemble voting across multiple neural network architectures and simulated model failures. Our

methodology encompasses the neural network models employed, the construction of ensembles, the introduction of failures via corrupted models, and the evaluation procedure. We focus on classification of 10-digit classes (0–9), a scenario where some models might be “blind” to certain digits to simulate partial failure.

3.1. Neural network models

We consider five types of neural network (NN) architectures of varying complexity:

SimpleLinear: A simple linear classifier (logistic regression) with no hidden layer, directly mapping 784 input features to 10 class scores. This model has 7,850 parameters (784×10 weights + 10 biases) and serves as a minimal baseline.

HiddenMLP: A multi-layer perceptron (MLP) with one hidden layer of 64 units and ReLu activation, adding non-linearity and capacity compared to SimpleLinear. The model maps 784 inputs to 64 hidden units and then to 10 class scores through the hidden layer for a total of 50,370 parameters.

SingleConv: A small convolutional neural network (CNN) with a single convolutional layer of 32 filters of size 3×3 , followed by flattening and a dense layer of 10 units. This model has approximately 3,074 parameters. It enables spatial feature extraction while keeping the network lightweight.

TinyCNN: A slightly deeper CNN with a single convolutional layer of 16 filters of size 3×3 and a dropout layer, followed by flattening and a dense layer of 10 units. This model has 1,498 parameters and achieves higher accuracy on digit classification due to its additional depth.

CompressedModel: A compressed MLP with a single hidden layer of 32 units and ReLU activation, mapping 784 inputs to 32 hidden units and then to 10 class scores through the hidden layer for a total of 25,370 parameters. This simulates scenarios where model size is constrained, potentially at some cost to accuracy.

Each architecture was trained on the digit classification task. We trained 17 *full* models per architecture, where “full” means the model was trained on the complete set of 10 digit classes (0–9). These 17 models were independently trained with different random initializations to provide diversity.

In addition to full models, we trained *corrupted* models to simulate omission failures. For each architecture, we trained models on datasets where 3 out of the 10 digit classes were omitted from the training set (the model never saw those classes and would likely misclassify them). We created four corruption schemes:

- **Scheme 0:** Models trained without digits $\{0, 1, 2\}$.
- **Scheme 1:** Models trained without digits $\{1, 2, 3\}$.
- **Scheme 2:** Models trained without digits $\{2, 3, 4\}$.
- **Scheme 3:** Models trained without digits $\{3, 4, 5\}$.

For each scheme, we trained seven models (28 corrupted models per architecture). The multiple models per scheme account for randomness in training and reduce bias from any single corrupted instance.

3.2. Ensemble construction

From the pool of 17 full models for each architecture, we constructed 100 ensembles, each consisting of 5 models. Each ensemble was created by randomly selecting 5 models out of the 17, ensuring diverse combinations.

While it is standard practice to balance ensembles by avoiding concentration of the best or worst models in the same group, in our case the performance deviations between models were so small that we omitted this step.

Each ensemble used *soft voting*: each model produced a probability distribution over the 10 classes (via softmax), and the probabilities were averaged element-wise. The predicted class was the one with the highest average probability. Soft voting naturally allows confident models to influence the decision while outliers are diluted by the consensus.

3.3. Simulating model failure (corruption schemes)

To evaluate robustness, we simulated single-model failures by replacing one model in an ensemble with a corrupted version of that model. For each of the 100 original ensembles, we generated four corrupted ensembles, one for each corruption scheme defined in Section 3.1.

Precisely one of the 5 ensemble members was swapped out, representing a single-point failure. The first model in each ensemble was replaced with a corrupted model from the same architecture's corrupted model pool. This yielded 100 corrupted ensembles per scheme, per architecture. All ensembles (both original and corrupted) employed the same soft voting mechanism.

3.4. Evaluation procedure

We evaluated every ensemble (and each corrupted variant) on a common test dataset of digit images specifically the MNIST dataset[9]. For each ensemble, we computed the metrics defined in Section 2: *Ensemble Accuracy*, *Ensemble Margin*, *Accuracy Drop*, and *Class Switching Probability (CSP)*. We also conducted a leave-one-out (LOO) analysis, in which each model was removed in turn to assess its impact on predictions.

The entire experimental pipeline – model training, ensemble construction, corruption injection, and evaluation – was implemented using TensorFlow/Keras. Trained models were saved and converted to the ONNX (Open Neural Network Exchange) format for interoperability with verification tools. ONNX conversion also enabled formal analysis using solvers by representing the ensemble as a single verifiable network with a soft-voting layer.

All metrics were computed from model predictions using Python scripts. We relied on `numpy` for statistical aggregation (e.g., averaging ensemble accuracies across 100 random ensembles per architecture and calculating standard deviations). When comparing original vs. corrupted predictions, we ensured the exact same test inputs were used and fixed all sources of randomness so that predictions were deterministic and reproducible.

Automated scripts generated random 5-model ensemble compositions from the pool of trained models and systematically injected corrupted models for each corruption scheme. Each original ensemble produced four corrupted versions (one per scheme), ensuring fair and uniform comparisons across all architectures.

All accuracy measurements, CSP calculations, and leave-one-out statistics were stored in structured logs and tables for traceability. This structured workflow enables the empirical results to align directly with the models used for formal verification, ensuring consistency between the experimentation and verification stages.

3.5. Formal verification on voting robustness

In order to encode NN models as a set of mathematical constraints, we rely on Marabou [8, 13]. The tool provides functionality for loading NN models from ONNX files and encoding them as a combination of linear equalities and inequalities, piecewise-linear constraints, and disjunctive constraints. Since Marabou does not support the encoding of Softmax layers due to their non-linearity, all Softmax layers must be removed in advance. Instead, we use the raw output values (logits) directly and encode the Argmax operation by introducing disjunctive constraints over the logits.

When loading multiple NN models and merging them into a single Marabou network, variable indexing must be handled carefully: (1) the input variables must be shared among all NNs, and (2) all other variables, including the outputs, must be *shifted* by an offset, computed on the fly, to avoid collisions in variable indices.

Encoding averaging for soft voting

To implement a soft voting scheme, the average of the output logits for each class must be computed. In Marabou, averaging can be encoded simply by summing the output variables and equating the result to a fresh variable – division by the number of models is unnecessary for verification purposes.

Let $n > 0$ be the number of NN models in the ensemble, and $m > 0$ the number of output classes. Let the variables o_{ij} denote the j th output logit of the i th NN model, where $1 \leq i \leq n$ and $1 \leq j \leq m$. We define s_j as a *fresh variable* representing the sum of logits for class j across all models:

$$\forall j \in \{1, \dots, m\} : \sum_{i=1}^n o_{ij} = s_j.$$

Encoding distinct argmax outputs

Let $s_1, \dots, s_m$ and $s'_1, \dots, s'_m$ denote the summed logits for two different ensembles (or models). We wish to encode the condition that the two ensembles predict *different* classes after applying Argmax. This can be expressed as the following disjunction:

$$\bigvee_{i,j=1}^n \left(\left(\bigwedge_{k=1}^n (s_i \geq s_k) \right) \wedge s'_i < s'_j \right).$$

Since Marabou supports only non-strict inequalities, the strict inequality $s'_i < s'_j$ must be rewritten as $s'_i \leq s'_j - \epsilon$, where $\epsilon > 0$ is a tunable constant controlling the precision for non-strict inequalities.

4. Experimental results

4.1. Experimental environment

All experiments were conducted on a cluster of 20 identical machines, forming a distributed computing setup. Each machine was equipped with an 8-core Intel(R) Xeon(R) W-2225 CPU at 4.10 GHz, 32 GB of RAM, and an NVIDIA RTX A4000 with 16 GB of VRAM. This setup allowed experiments to be parallelized across multiple machines. While this hardware configuration provided adequate computational resources for the neural network training phase and verification of simpler architectures, memory limitations became apparent during formal verification of more complex models.

The software environment consisted of TensorFlow 2.15 for training individual neural network models, NumPy for ensemble voting and statistical aggregation, Marabou 2.0 for formal verification encoding and constraint solving, and Gurobi 1.2.3 as the underlying optimization solver. The distributed setup enabled parallel execution of the 100 ensemble trials across multiple architectures and corruption schemes, significantly reducing the overall experimental runtime.

Following the training phase described in Section 3, all trained models were saved and subsequently converted to the ONNX (Open Neural Network Exchange) format to ensure compatibility with the Marabou verification framework. This conversion process maintained the mathematical equivalence of the models while enabling the constraint-based encoding required for SMT-based verification.

4.2. Empirical results

Now we present the empirical results demonstrating the impact of single-model corruption on ensemble performance over 100 random trials. Table 1 presents ensemble evaluation metrics for the different neural network architectures under both non-corrupted and corrupted conditions across four corruption schemes (S0, S1, S2, S3). The **Model** column represents the different neural network architectures. The **Corr.** column corresponds to the different corruption schemes and

their absence. The label “none” references the scenario with ensembles composed entirely of fully trained models, serving as a reference baseline, while S0...S3 correspond to ensembles where one model has been replaced by a corrupted model according to the specified corruption scheme. **Acc.** shows the average prediction accuracy over the trials. **Acc. Drop** indicates the decrease in accuracy relative to the baseline. **Margin** represents the average difference between the top two predicted class probabilities, quantifying prediction confidence. **CSP** (Class Switching Probability) measures how often the ensemble prediction changes when one member is removed or corrupted. **LOO Drop** quantifies the impact of leaving out a single model on ensemble accuracy.

4.2.1. Accuracy

As shown in the results, the accuracy did not vary significantly even when one model in the ensemble was replaced by a corrupted one. The largest observed accuracy drop is only 0.16% (see SimpleLinear in Table 1), demonstrating the robustness of our ensemble approach against individual model failures. This robustness is critical in maintaining reliable performance despite the presence of corrupted inputs.

4.2.2. Margin

While accuracy remained relatively stable, the ensemble margin consistently decreased under corruption, with the largest reduction being approximately 12%. Even the best performing model TinyCNN experienced a margin drop of roughly 11-12%. Nonetheless, the ensemble voting mechanism ensures that the corrupted model could not outweigh the consensus of the remaining four models, preserving decision stability.

4.2.3. Class Switching Probability (CSP)

The ratio of class switches, measured by the CSP, remained quite small; this indicates that our ensembles are stable under perturbations caused by corrupted models. CSP quantifies the proportion of samples for which an individual model alone changes the voting outcome. A low CSP implies that the corrupted model rarely influences the ensemble’s decision, enabling rapid SAT verification by the Marabou tool. High CSP values correlate with easier SAT input detection, while low CSP values correspond to more robust ensembles that resist corrupted influences.

4.2.4. Leave-One-Out (LOO) drop

The LOO drop, which measures the accuracy decrease when removing one model from the ensemble, reflects the accuracy drop seen with corrupted models. Smaller LOO drops indicate that even reduced ensembles perform comparably to the full ensemble, emphasizing the resilience of the voting mechanism. A large LOO drop signifies a model’s key role in correct decisions, meaning if corrupted, the ensemble is more likely to encounter SAT conditions. Conversely, small LOO drops suggest

Table 1. Evaluation results for different ensemble architectures (non-corrupted and corrupted).

Model	Corr.	Acc.	Acc. Drop	Margin	CSP	LOO Drop
SimpleLinear	–	0.92570	0.00000	0.84442	0.00000	0.00000
	S0	0.92430	0.00140	0.74527	0.00960	-0.00140
	S1	0.92410	0.00160	0.75020	0.01130	-0.00160
	S2	0.92450	0.00120	0.75910	0.01210	-0.00120
	S3	0.92460	0.00110	0.76557	0.00990	-0.00110
HiddenMLP	–	0.97430	0.00000	0.95354	0.00000	0.00000
	S0	0.97480	-0.00050	0.84248	0.00388	0.00050
	S1	0.97490	-0.00060	0.84143	0.00310	0.00060
	S2	0.97350	0.00080	0.84680	0.00380	-0.00080
	S3	0.97390	0.00040	0.85247	0.00420	-0.00040
SingleConv	–	0.98148	0.00000	0.97727	0.00005	-0.00001
	S0	0.98162	-0.00013	0.86241	0.00291	0.00013
	S1	0.98216	-0.00067	0.86128	0.00263	0.00067
	S2	0.98211	-0.00062	0.86461	0.00272	0.00062
	S3	0.98212	-0.00063	0.87166	0.00311	0.00063
TinyCNN	–	0.98232	0.00000	0.96820	0.00005	0.00000
	S0	0.98345	-0.00113	0.85642	0.00296	0.00113
	S1	0.98319	-0.00088	0.85450	0.00304	0.00088
	S2	0.98230	0.00002	0.85936	0.00332	-0.00002
	S3	0.98239	-0.00008	0.86576	0.00308	0.00008
CompressedM.	–	0.96500	0.00000	0.93244	0.00000	0.00000
	S0	0.96649	-0.00149	0.82676	0.00649	0.00149
	S1	0.96550	-0.00050	0.82351	0.00550	0.00050
	S2	0.96430	0.00070	0.83104	0.00570	-0.00070
	S3	0.96420	0.00080	0.83674	0.00560	-0.00080

that corrupted models have limited impact on voting outcomes, often resulting in UNSAT conditions. In our case, the LOO Drop is equal to the inverse of the accuracy drops of the corrupted ensembles, showing us that the smaller ensemble containing 4 full models is just as strong as our full 5 model ensemble.

Notable observations In some cases, ensembles containing a corrupted model appear to achieve slightly higher accuracy than the corresponding full ensemble. This behavior is likely due to the combination of two factors: (i) the MNIST dataset is relatively small and well-separated, so omitting a few classes in one model does not drastically affect predictions on the test set, and (ii) the corruption itself is mild, meaning the corrupted model may occasionally make correct predictions by chance that complement the other ensemble members. Consequently, random variations across the 100 ensemble trials can produce instances where the corrupted ensemble performs marginally better than the full ensemble. Importantly, these occurrences are rare and do not undermine the overall robustness trends observed across all metrics.

4.3. Formal verification results

We now present the verification results for the ensemble models described above. In our experiments, we used the Marabou solver to assess whether it could complete verification tasks within a timeout of 1200 seconds, and – most importantly – whether it could return a correct SAT (satisfiable) or UNSAT (unsatisfiable) result for different NN models under various corruption schemes.

For each NN model, we constructed 50 ensemble instances by randomly selecting 5 full models and 1 corrupted model. We then executed three distinct verification tasks on each ensemble:

Model Failure: Checks whether the full ensemble and a corrupted ensemble (in which the first intact model is replaced by the corrupted one) can produce different class predictions. If the verification result is SAT, then such inputs exist; if it is UNSAT, the two ensembles always agree on their prediction. In SAT cases, our tool also returns a set of input values as a counterexample.

Leave-One-Out (LOO): Checks whether the corrupted ensemble and the ensemble from which the corrupted model is entirely removed can predict different classes. The SAT/UNSAT interpretation is the same as in the Model Failure task.

Ensemble Equivalence: Checks whether two instances of the corrupted ensemble can produce different predictions. This task was implemented primarily to study how the solver handles UNSAT instances, as verifying UNSAT cases is generally considered more computationally demanding than verifying SAT cases.

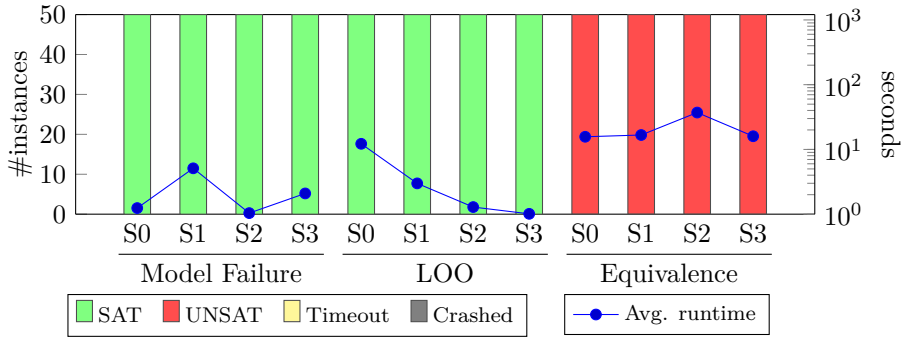
Table 2 provides insights into the scale of the verification problem instances solved for different types of ensemble models and verification tasks. The column **#Vars** denotes the number of decision variables, while **#ReLU**s represents the number of ReLU constraints. The **#Eqs** column reports the total number of equations and inequalities, followed by two statistical measures describing the number of addends in these constraints – **Mean** indicates the average number of addends, and **Median** shows the corresponding median value.

Figures 1, 2, and 3 present the verification results for the SimpleLinear, HiddenMLP, and CompressedModel ensembles, respectively. The bars are grouped according to the three verification tasks, with each group containing results for the four corruption schemes (S0, S1, S2, S3). The stacked bars display the distribution of SAT and UNSAT results, timeouts, and solver crashes across the 50 ensemble instances. The line plot shows the solver’s average runtime in seconds (note that the right-hand vertical axis is log-scaled).

As expected, most of the Model Failure and LOO instances are SAT, indicating that the ensembles are not completely robust with respect to their voting behavior. For these SAT cases, the solver successfully found counterexamples (adversarial inputs), even though these inputs were artificially crafted by the solver and are not part of the original dataset. However, some Model Failure and LOO instances were proven to be UNSAT, meaning that they are *absolutely robust* – the solver

Table 2. Statistics on the verification problem instances as modeled by Marabou.

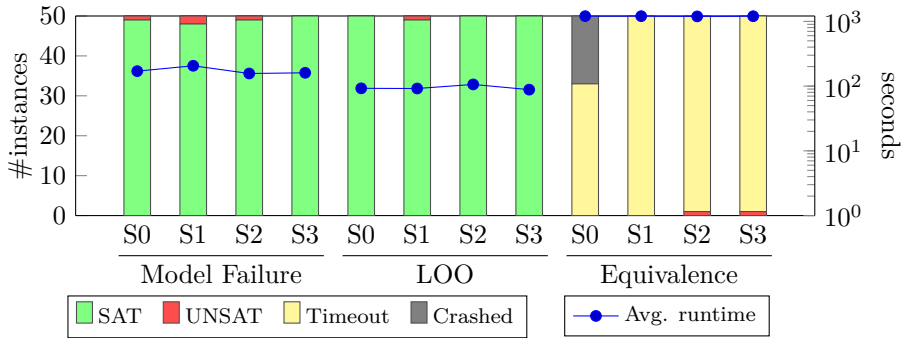
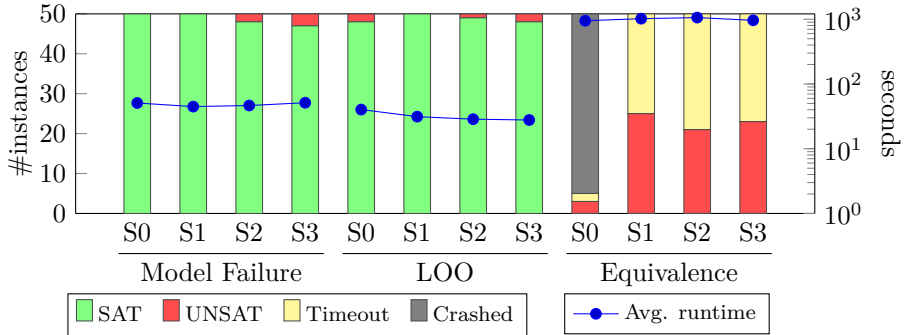
Model	Ver. Task	#Vars	#ReLUs	#Eqs	Mean	Median
SimpleLinear	Model Failure	864	0	80	590	785
	LOO	854	0	70	562	785
	Equivalence	844	0	60	655	785
HiddenMLP	Model Failure	1632	384	464	658	785
	LOO	1494	320	390	653	785
	Equivalence	1484	320	380	670	785
SingleConv	Model Failure	260448	129792	129872	20	10
	LOO	217174	108160	108230	20	10
	Equivalence	217164	108160	108220	20	10
TinyCNN	Model Failure	130656	64896	64976	20	10
	LOO	109014	54080	54150	20	10
	Equivalence	109004	54080	54140	20	10
CompressedM.	Model Failure	1248	192	272	562	785
	LOO	1174	160	230	554	785
	Equivalence	1164	160	220	579	785

Figure 1. Verification results for **SimpleLinear** ensembles.

was able to guarantee that no adversarial inputs exist.

Interestingly, for all SingleConv and TinyCNN ensembles, the solver crashed in every case due to the system running out of available RAM. We suspect that the solver was unable to handle the computational overhead introduced by convolutional layers in these models, though this issue requires further investigation. Similarly, we plan to investigate why the solver occasionally crashed when checking the equivalence of HiddenMLP and CompressedModel ensembles (see Figures 2 and 3).

We note that a few UNSAT instances were incorrectly reported as SAT by the solver. However, a manual review of the logs revealed that, in these cases, the two ensembles under investigation produced identical predictions. The apparent discrepancy arose from the encoding used for comparing Argmax outputs (see Sec-

Figure 2. Verification results for **HiddenMLP** ensembles.**Figure 3.** Verification results for **CompressedModel** ensembles.

tion 3.5), where the predicted values differed by less than the chosen precision threshold of $\epsilon = 0.001$.

5. Conclusion

Robustness in neural network ensembles is paramount for safety-critical AI applications, where high reliability is required even if some individual models fail. In this work, we introduce voting robustness as a measure of an ensemble's tolerance to model failures and demonstrate, through extensive experiments, that soft-voting ensembles can maintain accurate and stable predictions despite single-model corruption. Empirically, our results showed a negligible drop in accuracy (less than 1% in the worst case) when one network was compromised, confirming that the ensemble's performance degrades gracefully. No single member has undue influence on the final decision, and the consensus of the ensemble preserves overall reliability beyond what any standalone model could achieve.

Equally important, we combined empirical analysis with formal verification

to provide a rigorous assurance of robustness. Using an SMT-based verifier, we encoded the entire soft-voting ensemble into a single model and demonstrated that, in many cases, the ensemble’s prediction remains unchanged under any single-model failure. The formal analysis not only confirmed the ensemble’s resilience observed in testing but also identified corner-case adversarial inputs for certain ensembles, highlighting that empirical robustness alone does not guarantee absolute security. By uniting these approaches, our study offers both practical evidence and provable guarantees of robustness. This comprehensive evaluation enhances confidence in soft-voting ensembles and demonstrates the value of integrating empirical results with formal methods to verify the reliability of AI systems.

The challenges we encountered with current verification engines (e.g., solver memory exhaustion on convolutional networks and occasional precision-related errors) point to the need for more scalable and reliable neural network verifiers. Future verification frameworks should enhance their handling of complex architectures and large ensembles – for instance, by optimizing memory usage or incorporating problem-specific heuristics – so that formal certification can keep pace with the growing complexity of models. Furthermore, maximizing the vote margin and limiting the influence of any single model will not only improve empirical resilience but also facilitate formal verification.

Our study focused on single-model failure scenarios in a controlled classification task, and the SMT-based verification struggled with very deep or convolutional models due to scalability constraints. These limitations mark directions for future work. Exploring multiple simultaneous model failures, applying our framework to larger real-world datasets, and devising more efficient verification algorithms are natural next steps to generalize our approach. Despite these challenges, our work demonstrates a promising step toward neural network ensembles that combine high accuracy with provable robustness.

Code availability

The trained ensemble models, their ONNX representations, and all scripts used in the experiments are openly accessible at our Zenodo repository: <https://doi.org/10.5281/zenodo.17286858>.

References

- [1] L. ANTAL, E. ABRAHAM, H. MASARA: *Generalizing neural network verification to the family of piece-wise linear activation functions*, Science of Computer Programming 243 (2025), p. 103269.
- [2] R. BUNEL, I. TURKASLAN, P. H. TORR, P. KOHLI, P. K. MUDIGONDA: *A Unified View of Piecewise Linear Neural Network Verification*, in: Advances in Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., 2018, pp. 4795–4804, DOI: [10.48550/arXiv.1711.00455](https://doi.org/10.48550/arXiv.1711.00455).

- [3] R. EHLERS: *Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks*, in: Automated Technology for Verification and Analysis (ATVA), vol. 10482, Lecture Notes in Computer Science, Springer, 2017, pp. 269–286, DOI: [10.1007/978-3-319-68167-2_19](https://doi.org/10.1007/978-3-319-68167-2_19).
- [4] S. GOWAL, C. K. QIN, J. UESATO, T. MANN, P. KOHLI: *Improving Robustness using Generated Data*, Advances in Neural Information Processing Systems 34 (2021), pp. 4211–4224, DOI: [10.48550/arXiv.2110.09468](https://doi.org/10.48550/arXiv.2110.09468).
- [5] D. GROSS, N. JANSEN, G. A. PÉREZ, S. RAAIJMAKERS: *Robustness Verification for Classifier Ensembles*, in: Automated Technology for Verification and Analysis, 2020, pp. 271–287, DOI: [10.1007/978-3-030-59152-6_15](https://doi.org/10.1007/978-3-030-59152-6_15).
- [6] D. HENDRYCKS, K. ZHAO, S. BASART, J. STEINHARDT, D. SONG: *Natural adversarial examples*, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 15262–15271.
- [7] G. KATZ, C. BARRETT, D. L. DILL, K. JULIAN, M. J. KOCHENDERFER: *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks*, in: Computer Aided Verification (CAV), vol. 10426, Lecture Notes in Computer Science, Springer, 2017, pp. 97–117, DOI: [10.1007/978-3-319-63387-9_5](https://doi.org/10.1007/978-3-319-63387-9_5).
- [8] G. KATZ, C. BARRETT, D. L. DILL, K. JULIAN, M. J. KOCHENDERFER: *The Marabou Framework for Verification and Analysis of Deep Neural Networks*, in: Computer Aided Verification (CAV), Springer, 2019, pp. 443–452, DOI: [10.1007/978-3-030-25540-4_26](https://doi.org/10.1007/978-3-030-25540-4_26).
- [9] Y. LECUN, L. BOTTOU, Y. BENGIO, P. HAFNER: *Gradient-based learning applied to document recognition*, Proceedings of the IEEE 86.11 (1998), pp. 2278–2324.
- [10] T. PANG, K. XU, C. DU, N. CHEN, J. ZHU: *Improving adversarial robustness via promoting ensemble diversity*, in: International Conference on Machine Learning, PMLR, 2019, pp. 4970–4979.
- [11] G. SINGH, T. GEHR, M. MIRMAN, M. PÜSCHEL, M. VECHEV: *An Abstract Domain for Certifying Neural Networks*, in: Proceedings of the ACM on Programming Languages (POPL), vol. 3, 2019, pp. 1–30, DOI: [10.1145/3290354](https://doi.org/10.1145/3290354).
- [12] F. TRAMER, N. CARLINI, W. BRENDL, A. MADRY: *On adaptive attacks to adversarial example defenses*, Advances in neural information processing systems 33 (2020), pp. 1633–1645.
- [13] H. WU, A. OZDEMIR, A. ZELJIĆ, K. JULIAN, A. IRFAN, D. GOPINATH, S. FOULADI, G. KATZ, C. PASAREANU, C. BARRETT: *Parallelization Techniques for Verifying Neural Networks*, in: Formal Methods in Computer Aided Design (FMCAD), IEEE, 2020, pp. 128–137, DOI: [10.34727/2020/isbn.978-3-85448-042-6_20](https://doi.org/10.34727/2020/isbn.978-3-85448-042-6_20).

We are not afraid of the wolf! – AI usage attitudes among Hungarian informatics students

Gábor Kusper, György István Mátyás, Tamás Balla

Eszterházy Károly Catholic University
{[kusper.gabor](mailto:kusper.gabor@uni-eszterhazy.hu),[matyas.gyorgy.istvan](mailto:matyas.gyorgy.istvan@uni-eszterhazy.hu),[balla.tamas](mailto:balla.tamas@uni-eszterhazy.hu)}@uni-eszterhazy.hu

Abstract. We report a multi-instrument study of Hungarian informatics students' attitudes toward generative AI in programming education. Large Language Models (LLMs) are increasingly used to generate code, explain concepts, and support coursework, raising questions about reliability, skill development, and job security. Our study, *We Are Not Afraid of the Wolf!*, conducted at Eszterházy Károly Catholic University, combined six surveys across two waves with BSc students in Computer Science and Business Informatics. We tested three hypotheses: H1—students are not concerned that increasingly intelligent AI tools will hinder their job prospects; H2—better programmers use AI more effectively for programming tasks; H3—better programmers evaluate AI-generated code more critically. Results: H1 was partially supported—most view AI as a tool and express limited near-term concern, though medium-term uncertainty remains. H2 received partial support: programmer-quality proxies (course grade and self-assessment) showed weak-to-moderate positive associations with output in a 10-minute AI-assisted game development task. H3 was strongly supported: higher-competence students consistently review, debug, and seek to understand AI-generated code. Overall, students adopt a critical yet pragmatic stance: they leverage AI to increase efficiency while maintaining verification routines. The dataset is openly available and will be updated annually. So, there is new hope: higher education in informatics still makes sense, as in our results more skilled programmers outperform less skilled peers even when both use powerful AI tools.

Keywords: LLMs in Programming, AI in Computer Science Education, Student Attitudes Toward AI

AMS Subject Classification: 97P80 Artificial intelligence (educational aspects)

1. Introduction

When we asked Roland, a 2nd-year Computer Science BSc student, whether he was afraid of AI, his immediate reply was: “*Not at all!*” This unexpected answer made us wonder: perhaps today’s students approach AI with far less fear than our generation, which often views it with caution. This observation motivates our study: how do young computer science students perceive AI in their learning and future careers?

Roland’s response is not just an isolated remark but a starting point for a broader inquiry. To situate this question, we must consider the wider educational and technological context in which generative AI is emerging.

The rapid development of generative artificial intelligence (AI) has created new opportunities and challenges in computer science education. Large Language Models (LLMs), such as ChatGPT, GitHub Copilot, or Gemini, are increasingly used by students to generate source code, explain programming concepts, or assist in software development projects. While these tools provide immediate support and inspiration, they also raise questions about reliability, dependence, and their long-term effect on programming skills and job security.

The title of this paper, *We are not afraid of the wolf!*, reflects this debate. The “wolf” symbolizes the fear that AI may replace human programmers or reduce the value of their skills. Our research investigates whether Hungarian informatics students share this fear, or whether they instead approach AI with curiosity and acceptance. More specifically, the study explores how students majoring in Computer Science and Business Informatics at Eszterházy Károly Catholic University (EKCU) use AI in programming tasks, exam preparation, and creative projects, and how these experiences influence their self-confidence and career expectations.

The contribution of this paper is threefold. First, we describe our study design and data-cleaning methodology across two survey waves (total $n = 72 + 110$, predominantly second- and third-year Computer Science BSc students). Second, we evaluate the pre-defined hypotheses and report exploratory findings on relationships among AI usage, self-efficacy, creativity, and labor market attitudes. Third, we outline how other researchers can reuse our openly released datasets and bilingual codebook to facilitate replication and secondary analysis. We do not engage in a sustained ethical analysis here, even though the questionnaires included several items on this topic.

This study investigates three hypotheses: H1 (The Wolf): Informatics students are not concerned that increasingly intelligent AI solutions will make it harder for them to find jobs on the labor market. H2 (A New Hope): Good programmers are more effective at using AI solutions to complete programming tasks than less skilled programmers. H3 (An Old Style): Good programmers evaluate AI-generated code more critically.

The results of these hypotheses are presented in Sections 3 and 4. To support reuse and replication, we also make our dataset openly available at: <https://zenodo.org/records/17013486>.

1.1. Related work

Early work on large language models for code highlighted both their promise and limitations for program synthesis. DeepMind's *AlphaCode* showed that models trained on public code can solve competition-level problems, yet still produce fragile solutions [6]. Following this, research on GitHub Copilot examined usability, correctness, and productivity. User studies found that developers value Copilot for speeding up tasks but can struggle with understanding and debugging longer snippets [13]. Empirical evaluations further assessed correctness across benchmarks [7, 10], and classroom experiments reported substantial efficiency gains in specific tasks [14].

In computer science education, a growing body of work investigates whether LLMs help novices learn to program. Classroom studies suggest that access to ChatGPT can support task completion and perceived understanding, though outcomes vary with scaffolding and assessment design [14]. Systematic reviews indicate generally positive short-term effects on performance and motivation, coupled with concerns about reduced cognitive effort and the need for responsible integration [1, 3]. Further studies analyze students' attitudes toward AI-generated code and the detectability of such submissions in coursework [2, 4]. Recent surveys synthesize open challenges around reliability, assessment, and academic integrity in AI-assisted education [11].

In the Hungarian context, several strands of research highlight the diverse ways in which generative AI is entering education. A recent pilot study explored retrieval-augmented AI tutoring in higher education, demonstrating feasibility and strong learner engagement [9]. Earlier work examined the integration of AI into electronic learning environments, emphasizing its potential to extend e-learning infrastructures [8]. More recently, Toldi investigated adaptive learning enhanced by generative models [12], and Király raised concerns about the erosion of algorithmic thinking in the context of LLM use among computer science students [5]. Our study adds to this emerging body of Hungarian research by providing empirical, multi-instrument evidence from informatics undergraduates on AI usage patterns.

2. Methodology

This study investigates how undergraduate informatics students use and perceive generative AI in programming-related contexts. We combine questionnaires with short, task-based activities to address three predefined hypotheses:

- labor market attitudes (H1 – The Wolf),
- programming proficiency and effective AI use (H2 – A New Hope), and
- critical evaluation of AI-generated code (H3 – An Old Style).

Data were collected in two survey waves during regular classes and lab sessions at EKCÚ in Spring 2025. Participation was voluntary; informed consent and institutional ethics approval were obtained. Across the two waves, we obtained 72 and

110 responses, respectively, from second- and third-year BSc students in Computer Science and Business Informatics.

We administered six questionnaires across the two waves:

1. **General AI Attitudes** (Wave 1): broad, multi-block survey on AI use attitudes, self-efficacy, perceived overreliance and reliability, ethics, and labor market views.
2. **Game Development Pre** (Wave 1): prior experience, tool familiarity, learning goals, and flow antecedents.
3. **Game Development Post** (Wave 1): perceived speed, reliability, error-fixing effort, satisfaction, and creativity after a **40-minute** *Snake* implementation with AI support.
4. **Rust Pre** (Wave 1): confidence and intended AI strategies before an AI-tutored Rust learning session.
5. **Rust Post** (Wave 1): perceived tutor quality, learning, motivation, and confidence after the session.
6. **Consolidated H2 Questionnaire** (Wave 2): a single-form instrument capturing programmer quality and task performance in a **10-minute** minimalist *Snake* game implementation. The full questionnaire appears in Appendix A.

Unless noted otherwise, items used 5-point Likert scales; several binary items (Yes/No) and free-text questions were also included. In task settings, students were allowed to use AI (e.g., ChatGPT or Copilot) to the extent specified in the activity instructions.

The Wave 1 game task required implementing a minimalist *Snake* game within **40 minutes** (core mechanics mandatory; optional features such as score counters allowed). The Wave 2 task used an intentionally **10-minute** time box to elicit rapid AI-assisted development: students implemented as many minimal functions as possible and reported them both as a checklist and as a single total. The Rust activity followed a *pre-post* design with an AI tutor: the pre-questionnaire captured prior confidence and plans for AI use; the post-questionnaire captured perceived learning and tutor experience.

3. Results of the first wave

In this paper we present the results of two data collection waves. This chapter presents the result of the first one.

3.1. Data collection and cleaning

Survey-based analyses are sensitive to careless responding and incomplete records. Since our goal was to capture *motivated* respondents, we implemented a two-stage

cleaning strategy applied to all questionnaires (actual removals occurred in the General AI Attitudes survey and in Rust Pre):

- **Motivation filter (missingness):** rows were retained only if at least **50%** of the *numerically convertible* items (grades, Likert, averages, binary answers (e.g., Yes/No)) were answered. Free-text fields did not count toward this ratio.
- **Uniformity filter (entropy):** we computed the effective number of unique responses as $\exp(H)$, where H is the Shannon entropy (natural logarithm) of the respondent's numeric answer distribution. Binary items were coded as 1/0; free-text and other non-numeric values were ignored for these calculations. If $\exp(H) \leq 1.3$, the row was excluded, indicating near-constant responding (e.g., almost all 4s with occasional deviations).

This two-step filtering left **36 valid responses** out of 71 raw entries in the General AI Attitudes survey. For the other questionnaires, the same checks were run; only *Rust Pre* required exclusions due to uniformity, while the game-related and *Rust Post* datasets remained essentially unaffected.

The rationale behind this procedure was to exclude respondents who provided low-effort answers, either by skipping large parts of the survey or by mechanically repeating the same response option. By focusing on motivated participants, the analysis better reflects the genuine attitudes and behaviors of students toward AI usage.

Table 1. Data cleaning outcomes per questionnaire.

Questionnaire	Raw	Removed	Final	Primary reason
General AI Attitudes	71	35	36	Missing data
Game Development Pre	38	0	38	–
Game Development Post	37	0	37	–
Rust Pre	34	7	27	Uniformity
Rust Post	28	0	28	–

3.2. Analysis plan and validity

For scale-level reliability we computed Cronbach's α on Likert-type items (integer responses in $[1, 5]$). To handle missingness consistently across questionnaires, we applied a simple inclusion rule for reliability: keep items with $\geq 80\%$ non-missing responses and rows with $\geq 80\%$ non-missing across the selected items, then compute α on complete cases.¹

¹This avoids unstable estimates due to sparse item coverage; exploratory factor analysis is outside this paper's scope.

Table 2. Reliability summary per questionnaire (Likert items only; 80% item/row rule).

Questionnaire	Final n	Items used	Rows used	α
General AI Attitudes	36	10	36	0.605
Game Development Pre	38	7	38	0.142
Game Development Post	37	11	37	0.678
Rust Pre	27	8	27	0.765
Rust Post	28	9	28	0.696

The *Rust* questionnaires exhibit satisfactory internal consistency ($\alpha \approx 0.70$ – 0.77). The *Game Development Post* questionnaire shows acceptable reliability for exploratory analysis ($\alpha \approx 0.68$). The *General AI Attitudes* questionnaire’s reduced item count under the 80% rule still yields a moderate α (≈ 0.61), indicating heterogeneous but usable attitudinal indicators. The *Game Development Pre* block was intentionally diverse (experience, tools, motivation), which explains the low α ; we therefore treat it as a descriptive questionnaire rather than a single latent scale.

We report descriptive statistics (means, SDs, distributions) for all main variables. For predefined hypotheses we use Pearson correlation on Likert scores (common in education research); as a robustness check we replicate key correlations with Spearman’s ρ . For comparisons across AI-usage strata, we use two-sample tests: Welch t or Mann–Whitney, depending on normality/scale. All inferential analyses are computed on the cleaned datasets (Table 1).

Internal validity is limited by self-report and the correlational nature of several analyses. External validity is constrained by a single-institution Hungarian sample. Measurement validity is affected by short scales in some blocks and evolving AI tools. We mitigate these risks via transparent cleaning, a reproducible pipeline, reliability reporting, and robustness checks.

3.3. Results

All analyses use the cleaned General AI Attitudes sample ($n = 36$) and quality-checked task datasets. Before presenting detailed analysis for each hypothesis, we provide a brief overview.

- **H1 (The Wolf):** *Partially supported.* Students mostly view AI as a tool rather than a competitor, though some express medium-term labor market concerns.
- **H2 (A New Hope):** *Not decided in the first wave.* In the first wave, the planned pairing between programming quality and AI effectiveness could not be tested due to missing cross-questionnaire linkage.
- **H3 (An Old Style):** *Supported.* Higher-competence students critically evaluate AI-generated code through review, debugging, and understanding.

3.4. H1: The Wolf

The first hypothesis states: *H1 (The Wolf): Informatics students are not concerned that increasingly intelligent AI solutions will make it harder for them to find jobs on the labor market.*

H1 addresses labor market concerns: whether students fear that AI will reduce programming jobs, whether they see AI as a tool or a competitor, and whether they report direct experiences of job-search difficulties or dismissals.

On the item “*I see AI more as a tool than a competitor*”, students leaned strongly toward the tool perspective ($\bar{x} = 3.58$, $SD = 0.63$, $n = 26$). On the 5-point scale, **7.7%** disagreed (1–2), **26.9%** were neutral (3), and **65.4%** agreed (4–5). Expectations about AI “taking over” programming jobs varied by timeframe:

- **1–2 years:** $\bar{x} = 2.56$, $SD = 0.60$, $n = 18$; 1–2: **50.0%**, 3: **44.4%**, 4–5: **5.6%**.
- **3–4 years:** $\bar{x} = 3.09$, $SD = 0.83$, $n = 23$; 1–2: **30.4%**, 3: **30.4%**, 4–5: **39.1%**.
- **5–6 years:** $\bar{x} = 2.85$, $SD = 0.77$, $n = 26$; 1–2: **38.5%**, 3: **38.5%**, 4–5: **23.1%**.

Direct labor market experiences were rare: **15.2%** reported knowing a programmer who could not find employment ($n = 33$), and only **2.9%** reported knowing someone dismissed in the last 1–2 years ($n = 34$).

The frequency of AI use (e.g., asking an LLM to explain code) showed only weak, non-significant associations with attitudes: with the *tool vs. competitor* item $r = 0.09$ ($n = 24$), and with the *1–2 year takeover probability* item $r = -0.12$ ($n = 18$).

After entropy- and missingness-based filtering, the data indicate that students *do not generally fear* near-term job losses due to AI. Most frame AI as a tool, see a 1–2 year “takeover” as unlikely, and hold mixed expectations for 3–6 years. Direct negative experiences exist but remain in the minority. Overall, H1 is *partially supported*: students do not exhibit widespread short-term fears, yet medium-term uncertainty about AI’s labor market impact persists.

3.5. H2: A New Hope

The second hypothesis states: *H2 (A New Hope): Good programmers are more effective at using AI solutions to complete programming tasks than less skilled programmers.*

We used two indicators of programmer quality: the *theoretical programming exam grade* (High-level Programming 2, lecture) and a self-assessment item on programming competence. For AI effectiveness, four questionnaires were designed (*Game Development Pre/Post*, *Rust Pre/Post*) to capture speed, reliability, bug-fixing effort, satisfaction, and tutor quality.

Our intended primary test was to pair each student’s quality indicators with their task-level outcomes. However, the questionnaires were not linked by a common identifier: anonymity was prioritized, which prevented direct individual-level associations. As a result, only indirect proxies within single surveys could be analyzed (e.g., tool-vs-competitor framing, debugging practices, or post-task confidence). These showed weak and inconsistent relationships, making them insufficient for a decisive H2 test.

Rather than a simple limitation, this outcome highlights an important methodological lesson: anonymity safeguards must be balanced with the ability to test hypotheses across instruments. To address this, we designed a consolidated questionnaire; see Appendix A. This introduces an optional, privacy-preserving *call sign*. We used this questionnaire in the second wave; see Section 4.

3.6. H3: An Old Style

The third hypothesis states: *H3 (An Old Style): Good programmers evaluate AI-generated code more critically.*

H3 examines whether good programmers evaluate AI-generated code more critically.

Table 3 summarizes post-generation routines. A majority *review* AI code before running it ($\bar{x} = 3.45$, 55.3% select 4–5). Many report *fixing bugs themselves* ($\bar{x} = 3.15$), and an even larger share *try to understand* the generated code ($\bar{x} = 3.65$, 70.6% select 4–5). Consulting *documentation* is less frequent on average ($\bar{x} = 2.81$), and *restyling* to personal conventions sits in the mid-range ($\bar{x} = 3.35$). Systematic *cross-checking across multiple AI tools* is comparatively less common ($\bar{x} = 2.86$). Students report encountering errors in AI-generated code with moderate frequency ($\bar{x} = 3.37$), underlining the need for critical verification steps.

Table 3. Descriptive summary for H3 items (cleaned dataset). For 5-point items, percentages refer to response distribution within each item.

Item	n	Mean	SD	1–2%	3%	4–5%
Review AI code before running	38	3.45	0.69	10.5	34.2	55.3
I fix bugs myself after AI generation	52	3.15	0.75	21.2	42.3	36.5
Try to understand the generated code	17	3.65	0.61	5.9	23.5	70.6
Look up documentation to understand the code	43	2.81	0.73	37.2	44.2	18.6
Restyle the generated code to my conventions	46	3.35	0.71	13.0	39.1	47.8
Compare answers from multiple AI tools	42	2.86	0.75	35.7	42.9	21.4
How often do you find errors in AI-generated code?	68	3.37	0.90	13.2	42.6	44.1

Overall, students show a clear tendency toward *classical* quality assurance be-

haviors when working with AI-produced code: they review and actively try to understand it, often taking responsibility for bug fixing and, to a lesser extent, restyling. Documentation lookup and cross-tool triangulation are less common, which – together with the reported error frequency – suggests room for scaffolding (e.g., checklists or required code reviews) to ensure robust verification of AI outputs.

4. Results of the second wave: Is there a new hope?

This section presents the results of the second survey wave. It specifically addresses *H2 (A New Hope)* which was left undecided in the first wave, by using a single, consolidated questionnaire that enables direct pairing of programmer-quality indicators and task outcomes. The survey was completed by **110** participants, all **2nd- and 3rd-year Computer Science BSc** students. The full questionnaire appears in Appendix A.

4.1. Data collection and cleaning

The task was a time-boxed (10 minutes) *Snake* game implementation with AI support. We focused on two constructs: (i) **programmer quality** (A1, course grade; A2, self-assessed competence, both on 1–5 scales) and (ii) **task performance** recorded in Section C. Specifically, **C3** asked respondents to *tick which game features worked* at the end of the task – via a checklist that included the core mechanic (growth on dot; game over on wall/tail) and optional extras (e.g., score, accelerating snake, pick-up lives, obstacles, level switching, different dot types, sound/music, hall of fame), plus three other slots. We parsed this checklist into a numeric count, denoted *C3_count*. By contrast, **C4** requested a *single total*, the number of features implemented within ten minutes. Thus, *C3_count* derives from the marked features, while *C4* is the participant’s self-reported total. The full wording of C3–C4 appears in Appendix A.

The questionnaire also contained two manual timestamps (B5 and C1), but they proved unusable. The *Snake* programming task was introduced in the header of Section C; B5 was the last item before Section C, and C1 the first item within it. Both B5 and C1 asked participants to enter the current time in **hh:mm** format. We intended to compute time-on-task as C1–B5; however, in almost all responses the difference is only one minute, indicating that most participants read the task and immediately filled out C1 before beginning the work. Consequently, we did not use B5/C1 for cleaning.

Instead, we relied on C4, which records the self-reported number of functions implemented for the *Snake* game within 10 minutes using the chosen AI tool. To balance inclusiveness with engagement control, we defined two analysis cohorts based on C4:

- **Permissive cohort:** rows with non-numeric C4 were removed; C4 = 0 was dropped as non-engagement; all cases with C4 ≥ 1 were retained.

- **Strict cohort (“engagement filter”)**: same preprocessing but only cases with $C4 \geq 3$ were retained, reflecting the expectation that 2nd–3rd year BSc students using an LLM can implement at least three minimal functions in ten minutes.

4.2. Results on H2

We will see the following result in this chapter. **H2 (A New Hope):** *Partially supported.* Higher programmer-quality proxies show weak to weak-to-moderate positive associations with task performance in a 10-minute AI-assisted setting.

Analysis plan: We report Pearson correlations with Spearman’s ρ as a robustness check for ordinal/non-normal data. To bound the impact of the engagement filter, all primary associations are presented on both cohorts. Our focal question is whether A1 and A2 are positively associated with $C3_count$. All analyses use the consolidated second wave questionnaire and the cleaning rules above.

Analysis: Across the **Permissive cohort** ($C4 \geq 1$; $N = 104$ valid pairs), A1 showed a *weak, positive* trend with $C3_count$ (Pearson $r = 0.19$, $p = 0.052$; Spearman $\rho = 0.17$, $p = 0.081$), while A2 exhibited a *weak-to-moderate, positive* association (Pearson $r = 0.228$, $p = 0.020$; Spearman $\rho = 0.274$, $p = 0.0049$). In the **Strict cohort** ($C4 \geq 3$; $N = 86$), the A1– $C3_count$ trend persisted with similar magnitude (Pearson $r = 0.19$, $p = 0.078$; Spearman $\rho = 0.18$, $p = 0.099$), and A2– $C3_count$ remained positive (Pearson $r = 0.170$, $p = 0.118$; Spearman $\rho = 0.212$, $p = 0.050$). In short, stronger programmers (by grade and self-assessment) tended to list more implemented functions in the short AI-assisted task; effects were small and more clearly detectable for self-assessment (A2), especially in the permissive cohort.

Table 4. Summary of H2-related associations.

Pair	Cohort	N	r / ρ		p	Note
A1 $\leftrightarrow$ C3_count	Permissive ($C4 \geq 1$)	104	$r = 0.19$	$\rho = 0.17$	$p = 0.052 / 0.081$	Weak, positive trend
A1 $\leftrightarrow$ C3_count	Strict ($C4 \geq 3$)	86	$r = 0.19$	$\rho = 0.18$	$p = 0.078 / 0.099$	Similar magnitude
A2 $\leftrightarrow$ C3_count	Permissive ($C4 \geq 1$)	104	$r = 0.228$	$\rho = 0.274$	$p = 0.020 / 0.0049$	Weak-to-moderate, positive
A2 $\leftrightarrow$ C3_count	Strict ($C4 \geq 3$)	86	$r = 0.170$	$\rho = 0.212$	$p = 0.118 / 0.050$	Borderline ρ

4.3. Interpretation

Taken together, these results offer *partial support* for H2. In a short, AI-assisted programming task students with higher programmer-quality proxies (A1, A2) tend to achieve more implemented functions. The pattern is clearest for the self-assessment proxy (A2), while the grade proxy (A1) shows a consistent but weaker trend hovering near conventional significance thresholds. A reasonable reading is that the two proxies capture partly different aspects of “being a good programmer”:

the course grade reflects broader curricular achievement over time, whereas self-assessment may track task-immediate confidence and strategy use that matter when orchestrating AI assistance under time pressure.

Effect sizes are modest, which is unsurprising for at least three reasons. First, the task was intentionally brief (10 minutes), compressing performance variance. Second, our outcome measures trade precision for feasibility: *C3_count* aggregates a checklist of heterogeneous features, and *C4* is a single self-reported total. These two can diverge under time pressure, introducing measurement noise that typically attenuates observable associations. Third, differences in how students prompt, decompose, and verify AI outputs likely add further variance that our minimal instrument does not fully capture. Against this backdrop, the fact that the A2 signal remains detectable, especially in the permissive cohort, suggests a stable underlying tendency rather than a spurious fluctuation.

Reporting results on both a permissive and a strict cohort helps bound sensitivity to low-engagement cases. As expected, the strict cohort yields slightly higher average performance and reduced dispersion, yet the direction and relative strength of the A1/A2 associations persist. This stability implies that the observed tendencies are not driven solely by respondents with near-zero engagement. At the same time, we avoid over-interpreting the magnitude of the effects: the present design was optimized for short, classroom-feasible data collection rather than fine-grained performance measurement.

From a pedagogical perspective, the findings support a pragmatic view of AI in programming education. More skilled students appear to extract somewhat greater task-level benefits from the same class of AI tools, even in a tightly time-boxed setting. For instructors, this points to two complementary actions: (i) continue integrating AI workflows that reward strong problem-solving and code comprehension skills; and (ii) provide scaffolds (e.g., concise review checklists, debugging prompts, minimal test-driven steps) that help less skilled students translate AI outputs into working features more reliably.

In sum, within the constraints of a micro-task and minimalist measurement, we find that programmer-quality proxies relate positively – albeit modestly – to short-horizon AI-assisted output. This aligns with the broader narrative of the paper: AI tools can amplify productivity, but classical strengths in programming still matter.

4.4. Limitations and lessons

The principal instrumentation limitation was timing: because the time fields were positioned before task completion, we could not validate time-on-task. Future iterations should either place time entry at the end of the task block or, preferably, instrument an *automatic timer*. Likewise, manual enumeration of functions can diverge across C3 and C4 under time pressure; parsing and consistency flags help but do not eliminate noise. The consolidated instrument, available in Appendix A, will be deployed annually, and the datasets have been integrated into the *We are not afraid of the wolf!* open data release on Zenodo, enabling replication and

longitudinal analyses.

5. Open data release

Following ethics approval, we publish all datasets as *open-access* to support reuse, replication, and secondary analysis. Open data improves transparency, enables cumulative knowledge, and creates opportunities for new research beyond our own hypotheses (e.g., items on AI ethics, not analyzed here).

All materials are hosted on **Zenodo**, ensuring long-term preservation through OpenAIRE and CERN. The project landing page is at <https://zenodo.org/records/17013486>. Each release includes cleaned CSV files, metadata, and a bilingual (Hungarian–English) codebook. Updates will be issued annually with versioned DOIs, enabling both single-year and longitudinal analyses.

All data are anonymized and distributed under a CC BY 4.0 license. Users are asked to cite both the dataset DOI and this article.

Wave-specific releases:

First wave dataset: <https://zenodo.org/records/17013486>

Second wave dataset: <https://zenodo.org/records/17218128>

6. Conclusion and future work

This study offers a multi-instrument snapshot of how Hungarian informatics undergraduates perceive and use generative AI in programming. We find that students typically frame AI as a tool rather than a competitor and, consistent with classical software-engineering practice, higher-competence students tend to review, understand, and debug AI-generated code (supporting H3). While short-term labor market fears are limited, medium-term uncertainty remains (partial support for H1). A key limitation was our inability to decisively test H2 due to missing cross-form linkage; this was an anonymity-driven design error that we have remedied by introducing an optional, privacy-preserving call sign and a consolidated instrument, see Appendix A.

Beyond H1–H3, we formulated **Observation 1 (Prompt Productivity)**: issuing more prompts to the AI is associated with producing more working features within a tight, 10-minute time box. We observed positive associations between the number of prompts and task output, suggesting that rapid, iterative interaction can be beneficial in short-horizon, AI-assisted coding.

We also note **Observation 2 (Better Programmers Use Better Tools)** emerging from group-level patterns: more skilled programmers may gravitate toward, or extract more value from higher-yield AI tools. While suggestive, this requires controlled studies to disentangle self-selection from tool effects.

We plan to evaluate these observations in future work. We also plan to repeat the surveys in the upcoming academic years to build a larger dataset.

References

- [1] Y. ALBADARIN, M. SAQR, N. POPE, M. TUKIAINEN: *A systematic literature review of empirical research on ChatGPT in education*, Discover Education 3.1 (2024), p. 60, DOI: [10.1007/s44217-024-00138-2](https://doi.org/10.1007/s44217-024-00138-2).
- [2] C. K. Y. CHAN, W. HU: *Students' voices on generative AI: Perceptions, benefits, and challenges in higher education*, International Journal of Educational Technology in Higher Education 20.1 (2023), p. 43, DOI: [10.1186/s41239-023-00411-8](https://doi.org/10.1186/s41239-023-00411-8).
- [3] R. DENG, M. JIANG, X. YU, Y. LU, S. LIU: *Does ChatGPT enhance student learning? A systematic review and meta-analysis of experimental studies*, Computers & Education 227 (2025), p. 105224.
- [4] M. HOQ, Y. SHI, J. LEINONEN, D. BABALOLA, C. LYNCH, T. PRICE, B. AKRAM: *Detecting ChatGPT-generated code submissions in a CS1 course using machine learning models*, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, 2024, pp. 526–532, DOI: [10.1145/3626252.3630826](https://doi.org/10.1145/3626252.3630826).
- [5] S. KIRÁLY, E. TROLL: *Algorithmic Thinking at Risk? A Case Study on LLM Use Among Computer Science Students*, in: Proceedings of the International Conference on Formal Methods and Foundations of Artificial Intelligence (FMF-AI), Eszterházy Károly Catholic University, 2025, URL: <https://uni-eszterhazy.hu/fmf/m/abstracts>.
- [6] Y. LI, D. CHOI, J. CHUNG, N. KUSHMAN, J. SCHRITTWIESER, R. LEBLOND, T. ECCLES, J. KEELING, F. GIMENO, A. D. LAGO, T. HUBERT, P. CHOY, C. DE MASSON D'AUTUME, I. BABUSCHKIN, X. CHEN, P. HUANG, J. WELBL, S. GOWAL, A. CHEREPANOV, J. MOLLOY, D. J. MANKOWITZ, E. S. ROBSON, P. KOHLI, N. DE FREITAS, K. KAVUKCUOGLU, O. VINYALS: *Competition-level code generation with AlphaCode*, Science 378.6624 (2022), pp. 1092–1097, DOI: [10.1126/science.abq1158](https://doi.org/10.1126/science.abq1158).
- [7] A. MASTROPAOLO, L. PASCARELLA, L. PONZANELLI, M. TUFANO, G. BAVOTA: *On the Robustness of Code Generation Techniques: An Empirical Study on GitHub Copilot*, in: 45th IEEE/ACM International Conference on Software Engineering (ICSE), 2023, pp. 2149–2160, DOI: [10.1109/ICSE48619.2023.00181](https://doi.org/10.1109/ICSE48619.2023.00181).
- [8] G. MOLNÁR, Z. SZÜTS: *Use of artificial intelligence in electronic learning environments*, in: 2022 IEEE 5th International Conference and Workshop Óbuda on Electrical and Power Engineering (CANDO-EPE), IEEE, 2022, pp. 000137–000140.
- [9] R. NÉMETH, A. TÁTRAI, M. SZABÓ, P. T. ZALETNYIK, Á. TAMÁSI: *Exploring the use of retrieval-augmented generation models in higher education: A pilot study on artificial intelligence-based tutoring*, Social Sciences & Humanities Open 12 (2025), p. 101751.
- [10] N. NGUYEN, S. NADI: *An Empirical Evaluation of GitHub Copilot's Code Suggestions*, in: Proceedings of the 19th International Conference on Mining Software Repositories (MSR), 2022, pp. 1–5, DOI: [10.1145/3524842.3528470](https://doi.org/10.1145/3524842.3528470).
- [11] N. RAIHAN, M. L. SIDDIQ, J. C. SANTOS, M. ZAMPIERI: *Large language models in computer science education: A systematic literature review*, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, 2025, pp. 938–944.
- [12] L. TOLDI: *Generative AI-Enhanced Adaptive Learning: Integrating GPT Models for Personalized Content and Feedback*, in: Proceedings of the International Conference on Formal Methods and Foundations of Artificial Intelligence (FMF-AI), Eszterházy Károly Catholic University, 2025, URL: <https://uni-eszterhazy.hu/fmf/m/abstracts>.
- [13] P. VAITHILINGAM, T. ZHANG, E. L. GLASSMAN: *Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models*, in: CHI Conference on Human Factors in Computing Systems Extended Abstracts, 2022, DOI: [10.1145/3491101.3519665](https://doi.org/10.1145/3491101.3519665).

- [14] Y. XUE, Y. GUO, Y. JIA, Y. HUANG: *Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1*, in: Proceedings of the 46th International conference on software engineering: software engineering education and training, 2024, pp. 331–341, DOI: [10.1145/3639474.3640076](https://doi.org/10.1145/3639474.3640076).

A. Questionnaire on attitudes toward AI use in programming

This single questionnaire captures (A) baseline programmer quality, (B) general AI use attitudes in programming, (C) task-specific outcomes after a short coding exercise, (D) post-generation quality assurance attitudes, (E) attitudes toward AI assistance in programming, and (F) vibe coding experience.

Administration and ethics. The survey is part of a research study conducted at the EKC University Faculty of Informatics. Participation is entirely voluntary and anonymous. An ethics approval was granted by the EKC University Scientific Committee. *Note: This form is not optimized for mobile. Please fill it on a laptop/desktop with a development environment available.*

Please choose a private *call sign* you can remember (e.g., **AI-User11**). Use the *same* call sign whenever you fill in any similar form. Do *not* include personal data (name, nickname, birth date/age).

Unless stated otherwise, items use a 5-point Likert scale: *1 = Strongly disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly agree.*

Administrative header

ID1. Call sign (e.g., AI-User11): _____

A. Baseline programmer quality.

A1. *High-level Programming 2 (lecture) grade:* **1** (Fail) **2** **3** **4** **5** (Excellent) (select one)

A2. *By my own assessment, I am a good programmer.* (1–5)

B. General AI use attitudes in programming.

B1. *I use large language models (e.g., ChatGPT/Copilot) to generate code.* (1–5)

B2. *Using AI helps me complete programming tasks faster.* (1–5)

B3. *With AI, I finish more tasks within a fixed time.* (1–5)

B4. *AI use reduces the effort I spend on routine coding.* (1–5)

B5. *Please enter the current time (HH:MM), e.g., 11:23:* _____

C. Task-specific outcomes (to fill *after* the following short task). Short task (10 minutes, measure time precisely). *Develop a SNAKE game with AI assistance.*

Core function: The snake grows when it eats a dot; the game ends if it hits a wall or its own tail.

Possible extra features: score; accelerating snake; pick-up lives; obstacles; level switching; different dot types (e.g., speed-up/slow-down/bonus points); sound effects or background music; hall of fame; or any other *cool* feature.

C1. *Please enter the current time (HH:MM), e.g., 11:23: _____*

C2. *Which AI solution did you use for developing the game? _____*

C3. *Which features work (check all that apply)?*

- ☐ Core function (growth on dot; game over on wall/tail)
- ☐ Score
- ☐ Accelerating snake
- ☐ Pick-up lives
- ☐ Obstacles
- ☐ Level switching
- ☐ Different dot types (e.g., speed-up, slow-down, bonus points)
- ☐ Sound effects or background music
- ☐ Hall of fame
- ☐ Other cool feature #1.
- ☐ Other cool feature #2.
- ☐ Other cool feature #3.

C4. *Total number of features implemented in 10 minutes (select one): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12*

C5. *While developing the game, I fixed bugs myself. (1–5)*

C6. *Total number of prompts I issued to the AI: _____*

D. Post-generation quality assurance attitudes.

D1. *I review AI-generated code before running it. (1–5)*

D2. *I try to understand AI-generated code, not just execute it. (1–5)*

D3. *I fix bugs in AI-generated code myself. (1–5)*

D4. *I create (or generate) unit tests for AI-generated code (either before or after generation). (1–5)*

E. Attitudes toward AI assistance in programming.

E1. *I see AI more as a tool than a competitor.* (1–5)

E2. *After using AI, I feel more confident that my solution is correct and complete.*
(1–5)

F. Vibe coding experience.

F1. *I enjoyed working with AI support during the Snake development task.* (1–5)

F2. *I experienced “vibe coding” (a creative flow state with AI) while working on the task.* (1–5)

Evaluating profitability in sports betting using probabilistic models and betting strategies*

József Gergő Pál, Csaba Bíró

Eszterházy Károly Catholic University and Eötvös Loránd University
paljozsefgergo@gmail.com
biro.csaba@uni-eszterhazy.hu

Abstract. Sports betting has evolved into a multibillion-dollar global industry, raising the question of whether consumers can achieve sustainable long-term profit. This study explores whether combining probabilistic prediction models with various betting strategies can yield statistically significant profit in football betting. We examine six prediction models – including Poisson, logistic regression, Elo, Monte Carlo simulation, and two novel heuristics (Veto and Balance) – alongside five popular betting strategies: Flat Betting, Martingale, Fibonacci, Value Betting, and the Kelly criterion. A custom Python-based simulation system was developed using real match data from 539 unique football games played between March and May 2025. A total of 10,000 match groups were generated, each containing 25 unique matches, yielding *300,000 model-strategy runs* ($6 \times 5 \times 10,000$). Simulations preserved chronological order and modeled realistic stake adjustments. Our results highlight the complex relationship between predictive accuracy and profitability, and the limitations of exploiting statistical advantages in an efficient market. While some combinations showed short-term gains, consistent long-term profit remained elusive under most conditions. The findings provide insight into model performance, risk management, and the practical challenges of algorithmic sports betting. *This study is intended solely for academic purposes; the results should not be interpreted as practical betting advice.*

Keywords: sports betting, probabilistic models, Veto model, Balance model, Kelly criterion, value betting, simulation

AMS Subject Classification: 62P05, 91B84

*This research was supported by the Eköp-24 University Research Fellowship Program of the Ministry for Culture and Innovation from the Source of the National Research, Development and Innovation Fund.

1. Introduction

Sports betting, particularly football prediction, has become a global-scale phenomenon with both economic and social implications. The global betting market exceeded USD 240 billion in 2023 and is projected to reach nearly USD 350 billion by 2030 [7]. In the United States alone, the legal sector generated a record USD 13.7 billion in 2024 [6], while sportsbooks maintain stable profit margins of 9–10% [19].

Alongside its financial significance, sports betting also carries social risks, including problem gambling and addiction, particularly among young men [1, 13, 18]. These contrasting aspects – strong financial incentives versus societal challenges – make academic investigation into betting efficiency both timely and relevant. This paper addresses a central question: can probabilistic forecasting combined with structured betting strategies achieve sustainable long-term profit?

To explore this, we evaluate six prediction models: Poisson regression [14], Elo ratings [5, 10], Monte Carlo simulation [15, 16], logistic regression [9], and two novel heuristics developed for this study, the *Veto* and *Balance* models. These are tested with five betting strategies: flat betting, Martingale, Fibonacci progression, value betting, and the Kelly criterion [12].

The contribution of this work lies in identifying model–strategy combinations capable of outperforming bookmaker odds, while highlighting the roles of probability calibration, risk management, and market dynamics. Special emphasis is placed on the interpretability and performance of the proposed *Veto* and *Balance* models.

2. Related work

Research on sports betting has focused mainly on two areas: (1) probabilistic modeling of match outcomes and (2) betting market efficiency.

Early approaches were Poisson-based, starting with Maher [14] and later refinements such as Dixon–Coles [4] and Karlis–Ntzoufras [11]. The Elo rating system, originally for chess [5], has been adapted for football [10], while logistic regression models use team-level features to estimate outcome probabilities [8]. Monte Carlo simulations have been applied to account for uncertainty in match and season forecasts [16]. More recently, machine learning techniques have also been tested, though often limited by data sparsity and overfitting risks [2].

Market efficiency studies examined biases such as the favorite–longshot effect [17], expected value modeling [3], and the use of staking systems like the Kelly criterion [12]. However, relatively few works combined probabilistic modeling with automated betting simulations using real historical odds, and even fewer assessed actual profitability under multiple strategies.

To the best of our knowledge, no prior study has proposed the *Veto* or *Balance* models. These heuristic approaches aim to translate team form and balance into full 1X2 probability distributions with high interpretability and low computational cost.

Our work builds on the above foundations while specifically addressing whether custom-built models can achieve sustainable profit in realistic betting conditions.

3. Prediction models

We evaluate six probabilistic models for football prediction. Four are standard in the literature:

- **Poisson:** assumes independent Poisson goal distributions, with expected goals aggregated into 1X2 probabilities [14].
- **Monte Carlo:** simulates thousands of matches using Poisson sampling, deriving outcome frequencies [16].
- **Elo:** updates league-based ratings after each match and converts rating differences to probabilities using logistic functions [10].
- **Logistic Regression:** multinomial model trained on team-level features (e.g., shots, possession) with softmax normalization [8].

In addition, two novel form-based heuristics were developed for this study: the *Veto* and *Balance* models. Unlike the statistical or machine learning approaches above, these focus on recent team form and opponent strength to produce interpretable and computationally efficient forecasts.

Model Design and Naming Logic

- The **Veto model** is based on an asymmetric logic: the chance of a team winning is diminished by the strength of the opponent – hence, the opponent may “veto” the win.
- The **Balance model** represents a symmetric design: it averages a team’s performance with the weakness of the opposing team in equal proportion.

Definition 3.1 (Veto Model). Let n denote the number of recent matches considered for each team, and let `decay_factor` $\in (0, 1]$ be the exponential decay parameter. Define:

$$w_i = \text{decay_factor}^{n-i} \quad (\text{weight assigned to the } i\text{-th most recent match})$$

For a team T , define the exponentially weighted outcome probabilities:

$$P_T(\text{win}) = \frac{\sum_{i=1}^n w_i \cdot \mathbb{I}_{\text{win}}(i)}{\sum_{i=1}^n w_i},$$

$$P_T(\text{draw}) = \frac{\sum_{i=1}^n w_i \cdot \mathbb{I}_{\text{draw}}(i)}{\sum_{i=1}^n w_i},$$

$$P_T(\text{loss}) = \frac{\sum_{i=1}^n w_i \cdot \mathbb{I}_{\text{loss}}(i)}{\sum_{i=1}^n w_i}$$

where $\mathbb{I}_{\text{result}}(i)$ is the indicator function for result type.

Let H and A denote the home and away teams respectively. The raw match outcome probabilities are computed as:

$$\begin{aligned} P(1) &= P_H(\text{win}) \cdot (1 - P_A(\text{win})) \\ P(2) &= P_A(\text{win}) \cdot (1 - P_H(\text{win})) \\ P(X) &= \frac{P_H(\text{draw}) \cdot n_H + P_A(\text{draw}) \cdot n_A}{n_H + n_A} \end{aligned}$$

These values are then normalized:

$$P'(r) = \frac{P(r)}{P(1) + P(X) + P(2)} \cdot 100 \quad \text{for } r \in \{1, X, 2\}$$

Definition 3.2 (Balance Model). This model computes probabilities using symmetric averaging. For the same notation as above:

$$\begin{aligned} P(1) &= \left(\frac{P_H(\text{win}) + (1 - P_A(\text{win}))}{2} \right) \cdot 100 \\ P(2) &= \left(\frac{P_A(\text{win}) + (1 - P_H(\text{win}))}{2} \right) \cdot 100 \\ P(X) &= \left(\frac{P_H(\text{draw}) + P_A(\text{draw})}{2} \right) \cdot 100 \end{aligned}$$

Final normalization is applied:

$$P'(r) = \frac{P(r)}{P(1) + P(X) + P(2)} \cdot 100 \quad \text{for } r \in \{1, X, 2\}$$

Key Distinction Between the Two Models

- The **Veto model** is *asymmetric*: it suppresses a team's win probability if the opponent is also strong.
- The **Balance model** is *symmetric*: it applies equal weight to both teams' performance metrics in a balanced averaging approach.

Author's note These models were independently developed and offer novel heuristic approaches based on form-weighted probabilities. No similar implementation was found in existing literature.

4. Betting strategies

Five common strategies were tested for managing stake allocation:

- **Flat Betting:** Fixed stake per bet, regardless of confidence or odds. Serves as a baseline.
- **Martingale:** Doubles the stake after each loss to recover losses. High bankruptcy risk due to exponential growth.
- **Fibonacci:** Stake increases by the Fibonacci sequence after each loss. Slower than Martingale but still risky.
- **Value Betting:** Bet only when:

$$P \cdot \text{odds} > 1$$

indicating positive expected value. Uses a fixed stake.

- **Kelly criterion:** Stake fraction:

$$f = \frac{b \cdot P - (1 - P)}{b}, \quad \text{where } b = \text{odds} - 1$$

Maximizes long-term growth while avoiding overbetting.

Each strategy was applied consistently per model. Key metrics included total profit, bankroll evolution, and bankruptcy rate.

5. Simulation setup

A custom Python-based framework was developed to evaluate the combined performance of prediction models and betting strategies. It integrates a GUI, database storage, and API-Football data access, enabling interactive match selection, model execution, and result visualization.

Data. The dataset contained 539 unique matches played between March and May 2025. Teams were required to have sufficient historical statistics and valid 1X2 odds; matches with incomplete data were excluded.

Simulation Design. We generated *10,000 synthetic match groups* (25 matches each), yielding *300,000 model-strategy runs* ($6 \times 5 \times 10,000$) across six prediction models (Veto, Balance, Monte Carlo, Poisson, Logistic Regression, Elo) and five betting strategies (Flat, Martingale, Fibonacci, Value, Kelly criterion). Each run started with a bankroll of 10,000 units; fixed-stake betting used 1,000 units, while Kelly applied dynamic stake sizing. Value bets were placed only when $P \cdot \text{odds} > 1$.

Odds and Output. For each outcome, the best available bookmaker odds were used to approximate optimal odds shopping, providing an upper bound on theoretical returns. Simulation results included bankroll trajectories and detailed tables. Core components were unit-tested, and the full code is available at: https://github.com/JocmanHUN/Szakdolgozat-Pal_Jozsef_Gergo_NZ5MI3

Note. Optimal odds shopping is idealized and likely overestimates achievable ROI; in practice, account limits, taxes, and latency reduce effective profitability.

6. Results and discussion

The extensive simulation experiments presented insightful results on the intricate relationship between probabilistic model accuracy and betting strategy efficiency. While general profitability across all strategies remained challenging, significant variations emerged depending on specific model-strategy combinations. The key findings are detailed below.

Overall Predictive Performance. Table 1 presents a statistical summary of each model’s performance across 539 unique matches. While the Elo model achieved the highest accuracy, its lower average odds limited profitability. The Veto model produced competitive accuracy with higher win-odds, supporting its success under the Kelly criterion. In contrast, the Logistic Regression model, though less accurate, frequently selected high-return opportunities, contributing to its volatile performance.

Table 1. General model performance statistics based on 539 unique matches. The last two columns report the mean odds conditional on wins and losses, respectively.

Model	Avg. Odds	Correct Pred.	Accuracy (%)	Win. Odds	Loss Odds
Logistic Regression	2.91	218	40.45	2.34	3.29
Veto	2.80	225	41.74	2.34	3.12
Balance	2.58	238	44.16	2.13	2.94
Monte Carlo	2.48	254	47.13	2.12	2.80
Poisson	2.45	248	46.01	2.08	2.77
Elo	2.25	262	48.61	1.97	2.51

Detailed Analysis of Model Performances. Figure 1 illustrates all 10,000 bankroll trajectories produced by the Balance model under the Value Betting strategy. Despite the theoretical potential of the Balance model, the performance proved consistently weak across the simulations. Although some individual bankrolls showed moderate growth, the aggregate average was negative, demonstrating poor value identification. A plausible explanation for this behavior is the model’s inability to consistently select truly value-rich matches, leading to a frequent selection of marginal or negative expected-value bets.

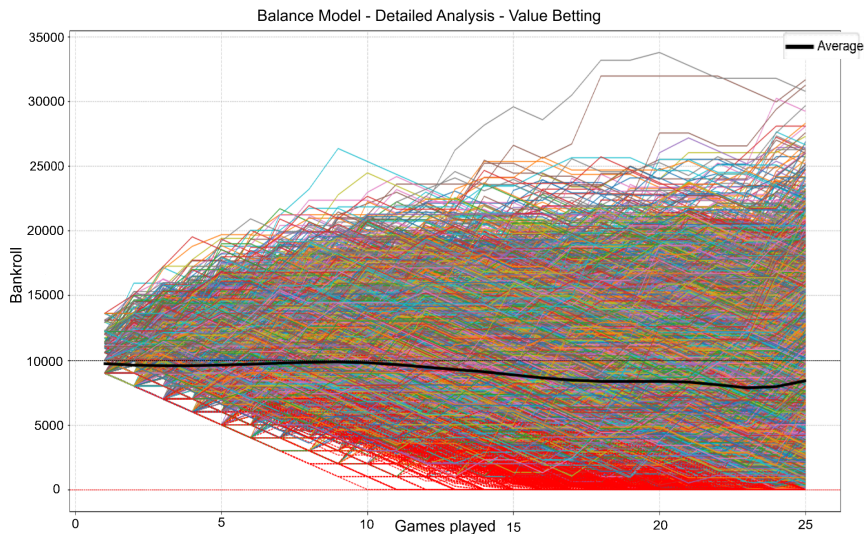


Figure 1. Detailed bankroll curves for the Balance model using Value Betting. All 10,000 simulations are shown; the black curve is the average bankroll.

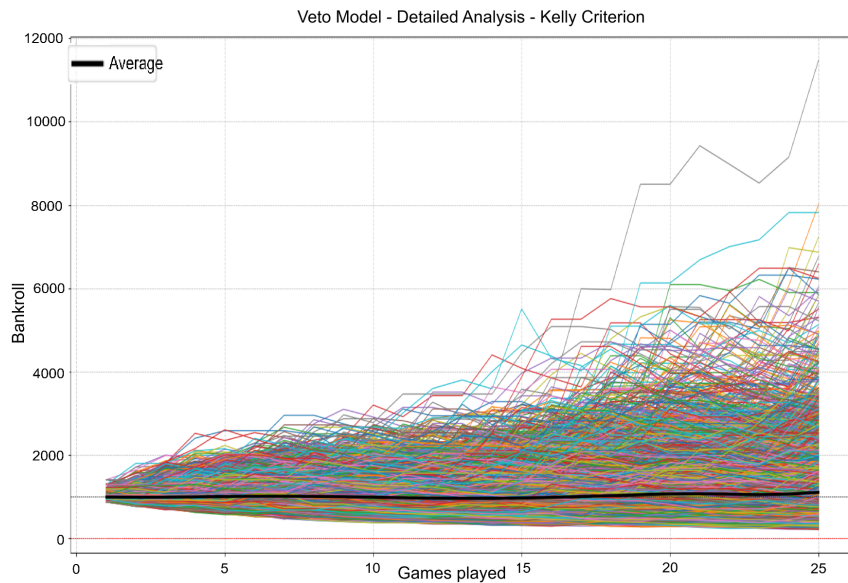


Figure 2. Detailed bankroll trajectories for the Veto model under the Kelly criterion. All 10,000 simulations are shown; the black curve is the average bankroll.

Contrastingly, the Veto model delivered remarkable results when combined with the Kelly criterion (Figure 2), which plots all 10,000 simulated bankroll paths alongside their average (black curve). This pairing stood out as the only consistently profitable combination across extensive testing. The exceptional outcome can be attributed to the model’s strong probability calibration, particularly within odds ranges most advantageous to Kelly-style proportional staking. Importantly, no bankruptcies occurred over 10,000 simulations, indicating both a stable risk profile and robust profitability potential. The success of the Veto model underscores the strength of heuristic-based approaches that incorporate recent team form and opponent quality into their calculations. Additionally, an important practical advantage of the Veto model is its extremely low computational resource requirements, allowing it to operate efficiently even on modest hardware.

ROI Analysis by Odds Range. To further investigate the Veto model’s success under the Kelly strategy, Table 2 breaks down the return on investment (ROI) by odds intervals. While the hit rate naturally declines with increasing odds, the highest ROI (+30.43%) was achieved in the 2.21–3.50 range – confirming that the model is particularly well-calibrated for identifying underpriced moderate-to-high odds. Surprisingly, very low odds (1.01–1.30) yielded perfect accuracy but only moderate ROI, while high odds consistently outperformed the others in profitability.

Table 2. ROI of the Veto model under Kelly strategy by odds range. Match Count counts only placed wagers (rounds where the Kelly fraction was zero are excluded). Bins were chosen to align with the model’s preferred odds corridor; alternative nearby binning yielded the same qualitative result.

Odds Range	Hit Rate	Match Count	ROI
Very Low (1.01–1.30)	100.00%	1,725	+23.74%
Low (1.31–1.60)	71.77%	4,602	-4.81%
Medium (1.61–2.20)	38.13%	33,641	-24.54%
High (2.21–3.50)	38.40%	76,017	+30.43%
Very High (3.51–10.00)	22.72%	73,144	-0.90%

The Logistic Regression model showed distinct characteristics under the Value Betting strategy, depicted in Figure 3. Its performance was marked by high volatility, frequent bankroll swings, and a considerable bankruptcy rate. All 10,000 simulated bankroll paths are plotted, with the black curve marking the mean trajectory. Nevertheless, the Logistic Regression was consistently able to detect matches where odds implied higher-than-actual risks, capitalizing on high-odds betting opportunities. The volatility is likely due to occasional overestimation of event probabilities, suggesting that while the model excels at identifying valuable bets, improved calibration or additional filtering mechanisms could stabilize its performance.

An overarching comparison of all models under the Kelly criterion (Figure 4)

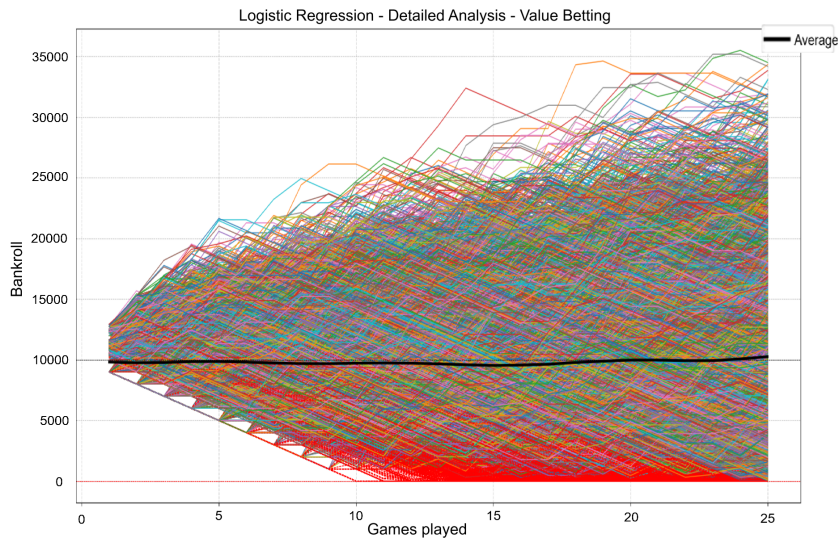


Figure 3. Detailed bankroll trajectories for the Logistic Regression model under Value Betting. All 10,000 simulations are shown; the black curve is the average bankroll.

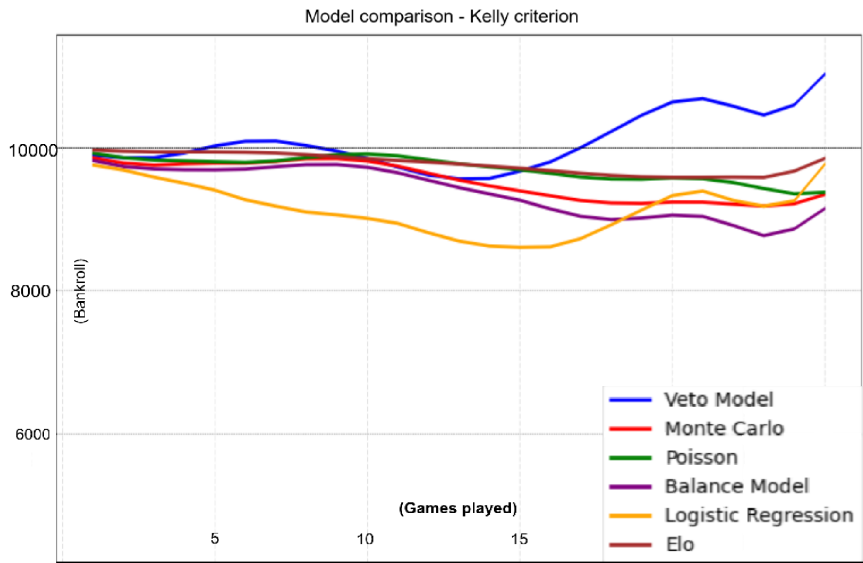


Figure 4. Comparative average bankroll performance of all models under the Kelly Criterion strategy.

plots the mean bankroll trajectory for each model; the Veto curve therefore coincides with the average line already shown in Figure 2, whereas the other curves depict the corresponding averages for their models. Unlike other models, which showed declining bankroll trends, the Veto model demonstrated a consistent upward trajectory, highlighting its superior long-term stability. The Elo model, despite having high prediction accuracy, suffered minor losses due to overly conservative odds selection, limiting its profitability potential. Models like Monte Carlo and Poisson showed moderate losses, indicating reasonable calibration but insufficient precision for consistently profitable outcomes.

It is essential to consider several practical limitations inherent to this research. The study's timeframe (March–May) was strategically chosen due to the density of competitive matches, yet following this period, notably fewer matches are available during the summer months, potentially altering model performance due to seasonal effects. Furthermore, many matches had to be excluded entirely due to missing or incomplete statistical data, limiting the total predictive coverage achievable in real-world scenarios.

Another critical assumption within the simulation was the use of optimal odds shopping, wherein the best available odds were always selected. However, in a practical environment, odds shopping is significantly more complex, as bettors must continuously monitor multiple bookmakers, navigate various platforms, and handle differing national regulations and taxes, all of which may substantially reduce realized profitability. Moreover, executing a successful betting strategy – especially strategies relying on dynamically adjusting stakes such as Kelly – requires constant attention and rapid decision-making. This implies a significant time and logistical commitment from bettors, which might not be feasible for every participant in real-world settings.

Key Insights on Strategy Efficiency. The simulations reaffirmed the critical importance of choosing betting strategies compatible with the predictive model's inherent risk profile and calibration characteristics. Aggressive strategies like Martingale and Fibonacci exhibited high bankruptcy rates across all models, indicating their unsuitability for sustainable long-term use. Conversely, the Kelly criterion and Value Betting strategies demonstrated clear potential, but only when aligned with appropriately calibrated models. Specifically, the Veto model's compatibility with Kelly criterion betting points towards a best-practice combination for maximizing returns while minimizing financial risk, albeit with the previously mentioned practical limitations considered.

7. Conclusion

This research sought to rigorously evaluate the profitability potential of combining probabilistic sports betting models with well-known betting strategies. Through extensive computational experiments involving 10,000 simulations per strategy-model combination, several critical conclusions emerged:

Predictive Accuracy and Odds Calibration. All models demonstrated statistically significant predictive skill compared to random guessing (33.33%). Elo ratings produced the highest accuracy (48.61%), closely followed by Monte Carlo, Poisson, and Balance models. Despite slightly lower accuracy rates, the Veto and Logistic Regression models effectively targeted higher odds, crucial for profit generation under specific strategies.

Sustainable Profitability. Achieving long-term profitability proved challenging for most model-strategy pairs. The notable exception was the Veto model combined with the Kelly criterion strategy, consistently generating positive average returns (+10.17%) without bankruptcy occurrences. This underscores the paramount importance of probability calibration and disciplined stake sizing in betting scenarios. The Logistic Regression model showed profitability potential through Value Betting but required improved volatility management.

Influential Factors. Several factors emerged as essential to successful betting systems:

- **Probability Calibration:** Precise probability estimation was critical. Poor calibration quickly eroded bankrolls, especially under strategies like Kelly, which heavily penalize inaccuracies.
- **Risk Management:** Strategies incorporating dynamic stake adjustment (e.g., Kelly) demonstrated significant resilience against bankroll depletion, highlighting the importance of adaptive risk management.
- **Selective Odds Range Betting:** Models excelling within particular odds ranges, such as Veto in moderate-odds markets and Logistic Regression in high-odds markets, significantly benefited from tailored betting strategies.

Significance of the Proposed Models. The study introduced two original heuristic-based models: Veto and Balance. While Balance underperformed, primarily due to insufficient value detection and overly concentrated betting suggestions, the Veto model exhibited outstanding performance. Its combination of exponentially weighted recent form and asymmetric opponent strength assessments provided exceptional calibration and robust predictive reliability. This highlights that even relatively simple heuristics can compete effectively with more sophisticated statistical models when carefully calibrated and appropriately applied.

Future Research Directions. Future extensions could significantly enhance model accuracy and practical usability:

- Integration of advanced machine learning algorithms (e.g., ensemble methods, neural networks, XGBoost).

- Incorporation of additional predictive factors such as player injuries, lineup changes, weather conditions, or betting market dynamics.
- Expanding historical datasets to strengthen statistical reliability.
- Enhancing computational resources via parallel computing or cloud-based systems.
- Development of a public-facing web-based betting advisory platform based on refined versions of these models.

These extensions could further validate whether the identified Veto–Kelly combination maintains profitability in broader and more realistic contexts.

Final Remarks. This work clearly demonstrates the challenges inherent in generating consistent profits from sports betting but also highlights viable pathways towards sustainable profitability through rigorous model calibration, strategic betting approaches, and disciplined risk management. The exceptional performance of the Veto model within the Kelly criterion framework serves as a compelling proof-of-concept, emphasizing that carefully designed probabilistic models, even heuristic-based, can yield meaningful advantages in competitive betting environments.

Ethical note. This work is for academic analysis only. Sports betting carries risks, including gambling addiction; results here should not be construed as betting advice, and real-world frictions (limits, taxes, latency) further reduce practical applicability.

References

- [1] M. ABBOTT, U. ROMILD, R. VOLBERG: *The prevalence, incidence, and gender and age-specific incidence of problem gambling: Results of the Swedish longitudinal gambling study (Swelogs)*, *Addiction* 113.4 (2018), pp. 699–707, DOI: [10.1111/add.14083](https://doi.org/10.1111/add.14083).
- [2] M. CARMONA-FREIRE, M. SORIANO: *A machine learning approach to model the outcome of football matches*, *IEEE Access* 8 (2020), pp. 118713–118719, DOI: [10.1109/ACCESS.2020.3004686](https://doi.org/10.1109/ACCESS.2020.3004686).
- [3] D. CORTIS: *Expected values and variance in bookmaker payouts: A theoretical approach towards setting limits on odds*, *Journal of Prediction Markets* 9.1 (2015), pp. 1–14, DOI: [10.5750/jpm.v9i1.987](https://doi.org/10.5750/jpm.v9i1.987), URL: <http://ubplj.org/index.php/jpm/article/view/987>.
- [4] M. J. DIXON, S. G. COLES: *Modelling association football scores and inefficiencies in the football betting market*, *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 46.2 (1997), pp. 265–280, DOI: [10.1111/1467-9876.00065](https://doi.org/10.1111/1467-9876.00065).
- [5] A. E. ELO: *The Rating of Chessplayers, Past and Present*, London: Batsford, 1978.
- [6] ESPN: *U.S. sports betting industry posts record \$13.7B revenue in 2024*, https://www.espn.com/espn/betting/story/_/id/43922129/us-sports-betting-industry-posts-record-137b-revenue-24, Hozzáfértés ideje: 2025-04-23, 2024.

- [7] GRAND VIEW RESEARCH: *Online Gambling Market Size, Share & Trends Analysis Report*, Letöltve: 2024. április 5., 2023, URL: <https://www.grandviewresearch.com/industry-analysis/online-gambling-market>.
- [8] A. GROLL, C. LEY, G. SCHAUBERGER, H. VAN EETVELDE: *Prediction of football match outcomes: a comparison of predictive models and the role of social factors*, International Journal of Forecasting 34.3 (2018), pp. 366–378, DOI: [10.1016/j.ijforecast.2018.02.006](https://doi.org/10.1016/j.ijforecast.2018.02.006).
- [9] D. W. HOSMER, S. LEMESHOW, R. X. STURDIVANT: *Applied Logistic Regression*, 3rd, Hoboken, NJ: Wiley, 2013, DOI: [10.1002/9781118548387](https://doi.org/10.1002/9781118548387).
- [10] L. M. HVATTUM, H. ARNTZEN: *Using ELO ratings for match result prediction in association football*, International Journal of Forecasting 26.3 (2010), pp. 460–470, DOI: [10.1016/j.ijforecast.2009.10.002](https://doi.org/10.1016/j.ijforecast.2009.10.002).
- [11] D. KARLIS, I. NTZOUFRAS: *Analysis of sports data by using bivariate Poisson models*, Journal of the Royal Statistical Society: Series D (The Statistician) 52.3 (2003), pp. 381–393, DOI: [10.1111/1467-9884.00366](https://doi.org/10.1111/1467-9884.00366).
- [12] J. L. KELLY: *A New Interpretation of Information Rate*, Bell System Technical Journal 35.4 (1956), pp. 917–926, DOI: [10.1002/j.1538-7305.1956.tb03809.x](https://doi.org/10.1002/j.1538-7305.1956.tb03809.x).
- [13] H. LOPEZ-GONZALEZ, M. D. GRIFFITHS: *Understanding the convergence of markets in online sports betting*, International Review for the Sociology of Sport 53.7 (2018), pp. 807–823, DOI: [10.1177/1012690216680602](https://doi.org/10.1177/1012690216680602).
- [14] M. J. MAHER: *Modelling association football scores*, Statistica Neerlandica 36.3 (1982), pp. 109–118, DOI: [10.1111/j.1467-9574.1982.tb01597.x](https://doi.org/10.1111/j.1467-9574.1982.tb01597.x).
- [15] N. METROPOLIS, S. ULAM: *The Monte Carlo Method*, Journal of the American Statistical Association 44.247 (1949), pp. 335–341, DOI: [10.1080/01621459.1949.10483310](https://doi.org/10.1080/01621459.1949.10483310).
- [16] G. PANTUSO, F. LERA-LOPEZ: *Monte Carlo simulation in sports betting: a football application*, Journal of Sports Analytics 3.3 (2017), pp. 161–172, DOI: [10.3233/JSA-170241](https://doi.org/10.3233/JSA-170241).
- [17] D. PEEL, D. THOMAS: *Betting market efficiency and the favourite-longshot bias: A survey of the literature*, Bulletin of Economic Research 67.1 (2015), pp. 42–57, DOI: [10.1111/boer.12035](https://doi.org/10.1111/boer.12035).
- [18] A. M. RUSSELL, N. HING, M. BROWNE: *Risk factors for gambling problems specifically associated with sports betting*, Journal of Gambling Studies 35.4 (2019), pp. 1211–1228, DOI: [10.1007/s10899-019-09848-x](https://doi.org/10.1007/s10899-019-09848-x).
- [19] S&P GLOBAL: *American Gaming Association: Legal sports betting hits record revenue in 2023*, <https://www.spglobal.com/market-intelligence/en/news-insights/articles/2024/2/american-gaming-association-legal-sports-betting-hits-record-revenue-in-2023-80522087>, Hozzáférés ideje: 2025-04-23, 2024.

cRAMI 4.0 an Improved Reference Architectural Model for Industrie 4.0 (RAMI 4.0) based on a three-dimensional cubic lattice model

Ioan Sima^{a*}, Daniela-Maria Cristea^{bc},
Elisabeta-Mihaela Ciortea^b, Laszlo Barna Iantovics^d

^aComputer Science Department, Babes-Bolyai University, Romania
ioan.sima@ubbcluj.ro

^bInformatics, Mathematics and Electronics Department,
University ‘1 Decembrie 1918’ of Alba Iulia, Alba-Iulia, 510009, Romania
daniela.cristea,mciortea@uab.ro

^cDoctoral School of Letters, Humanities and Applied Sciences,
George Emil Palade University of Medicine, Pharmacy, Sciences and Technology of
Targu Mures, 540142
cristea.daniela-maria.23@stud.umfst.ro

^dElectrical Engineering and Information Technology Department,
George Emil Palade University of Medicine, Pharmacy, Sciences and Technology of
Targu Mures, 540142, Romania
barna.iantovics@umfst.ro

Abstract. In the context of Industry 4.0, the increasing complexity of 5G-enabled Internet of Things (IoT) systems demands efficient and scalable methods for optimizing information flow across distributed architectures. This paper presents a novel approach to optimizing information flows in IoT systems by leveraging a three-dimensional cubic lattice model aligned with a customized Reference Architectural Model for Industry 4.0 (cRAMI 4.0). The novelty and improvement consist in the use of a three-dimensional lattice structure that maps three critical axes, detection level functions (X-axis), analysis and maintenance processes (Y-axis), and access control levels (Z-axis), onto a unified spatial model for representing and optimizing in-

*Corresponding author.

formation flows in IoT systems. Each node within the lattice represents a combination of these dimensions, enabling a comprehensive representation of functional, procedural, and security aspects in IoT environments. This spatial model facilitates the visualization, analysis, and optimization of data flows and access control mechanisms, enhancing system efficiency, security, and maintainability. The approach offers practical benefits for designing scalable and secure IoT architectures compliant with the main RAMI 4.0 principles. We formalise the proposed system using Petri net based modelling to simulate process transitions across Cloud-IoT-Lattice domains. Each IoT layer (perception, network, application) is treated as a distinct computational zone, and transitions between zones are defined across lattice-encoded relations. This interdisciplinary approach bridges informatics and industrial informatics, offering a new paradigm for adaptive, low-latency IoT systems in smart manufacturing, healthcare, and logistics.

Keywords: Internet of Things, RAMI 4.0, cRAMI 4.0, cubic lattice model, Petri nets

1. Introduction

The concept of the *Internet of Things* (IoT) describes a highly interconnected world, in which various objects are integrated with sensors and other digital devices, enabling their networking for the collection and exchange of data [19]. These devices, often referred to as *connected objects* or *IoT devices*, range from household appliances to industrial machines, and are equipped with sensors and actuators to gather and transmit data [8].

In the context of manufacturing, IoT enables real-time data collection and sharing among various production resources, such as machines, workers, materials, and tasks. Furthermore, IoT can provide enhanced connectivity across objects, systems, and services, allowing data exchange [19].

In the future, a convergence of IoT-related technologies is anticipated, including ubiquitous wireless standards, Data Analytics, and Machine Learning (ML) [19]. Industrial IoT (IIoT) is considered a key driving force for increasing productivity and efficiency in industrial landscapes. IoT is one of the core technologies underpinning Industry 4.0 [8].

The *Reference Architectural Model Industrie 4.0* (RAMI 4.0) is widely regarded as the preferred framework for implementing Industry 4.0 architectures [9, 15, 18]. It is represented as a three-dimensional cube encompassing the most important business elements and technological innovations of Industry 4.0. RAMI 4.0 aims to guide the implementation of compatible system architectures and to provide a shared framework for stakeholders to understand and communicate effectively. The model facilitates the classification of objects such as machines, and the description and implementation of complex Industry 4.0 (I4.0) concepts.

Regarding the *integration* of IoT within the RAMI 4.0 structure, the RAMI provides a framework for incorporating information technology/operational technology (IT/OT) systems (in which IoT plays a crucial role) into industrial ecosystems.

Those three axes of RAMI 4.0 (hierarchical levels, life cycle stages, and architecture layers) allow mapping and analysis of how IT/OT microsystems interact and contribute to value creation. The 3D visualisation of microsystems within RAMI 4.0 cubes demonstrates how legacy IoT systems can be integrated with newer I4.0 technologies by treating legacy systems as *closed systems*. Mapping administrative shells and technical assets to appropriate layers ensures the functional and communication properties are correctly assigned [13].

Petri nets [1] have long been employed as a formal modeling tool for the analysis of discrete event systems, particularly in the context of concurrency, synchronization, and resource sharing [4, 16, 21]. Due to their graphical nature and rigorous semantics, Petri nets are especially suited for representing the behavior of distributed systems, making them a natural fit for modeling Internet of Things (IoT) environments. In recent years, their use has expanded into Industry 4.0 contexts, where complex device interactions and process flows require formal verification and simulation tools.

Lattice structures are widely used in computer science and engineering to represent discrete, multi-dimensional spaces in which elements are organized based on defined rules of adjacency and interaction. In the context of system modeling, a lattice provides a structured way to map complex relationships across multiple dimensions, enabling clear visualization and analysis of interdependencies. Particularly in domains where hierarchical, procedural, and control aspects intersect - such as in industrial IoT systems - a lattice can serve as an effective abstraction for organizing system components and their interactions. By defining nodes at the intersection of key dimensions, lattice-based models allow for both conceptual clarity and computational tractability in modeling, simulation, and optimization tasks.

The innovative contribution of this work lies in the integration of a three-dimensional cubic lattice model with a customized RAMI 4.0 (cRAMI 4.0) framework, where the axes correspond to detection level functions, analysis and maintenance processes, and access control levels. This structural approach enables a new perspective on modeling and optimizing information flows in IoT systems. Based on this architectural model can be described agent-based systems in Industry 4.0 developments [10, 11].

To capture the dynamic and often unpredictable nature of IoT environments, we extend our modeling approach to include both deterministic and stochastic Petri net simulations. While deterministic models provide a baseline for system behavior under ideal conditions, stochastic modeling allows us to incorporate variability in transition delays, fault occurrences, and data flow rates. This dual perspective enables a more realistic evaluation of system responsiveness, fault tolerance, and access control enforcement, especially in scenarios involving fluctuating sensor inputs, network latency, or probabilistic failures.

The following sections detail the theoretical foundation of the model, its mapping to the custom RAMI axes, and the simulations with experimental evaluation purposes using Petri Nets.

2. Reference Architectural Model Industry 4.0

Internet of Things (IoT) refers to the network of interconnected physical devices that can collect, transmit and process data from the real world (sensors, actuators, smart devices, etc.) [17]. In the following, we will highlight several aspects concerning data analysis in IoT systems and their architectures [8].

Data Analysis Challenges in IoT. These include the vast volume and variety of data, the need for real-time processing and long-term maintenance.

IoT Data Processing Techniques. Techniques such as data denoising, outlier detection, missing data imputation, and data aggregation are frequently employed.

Integration with Emerging Technologies. Data analysis in IoT is increasingly integrated with cloud computing, fog computing, and edge computing to address specific challenges in sensor networks and data analytics.

Typical IoT Architectures. These include multiple layers: the Perception Layer (sensors and actuators), the Network Layer (routers and gateways, connectivity and protocols), the Application Layer (cloud/servers, industrial applications), and occasionally a middleware or support layer for intelligent data processing and decision-making [8].

The RAMI 4.0, proposed for understanding and implementing Industry 4.0, highlights the importance of security as a fundamental condition.

RAMI 4.0 plays a central role in structuring and providing a common understanding of the complex systems specific to Industry 4.0 [22]. Its main roles include the following:

- **Developing a shared understanding:** The main goal of RAMI 4.0 was to guarantee that all parties involved understood Industry 4.0.
- **Structuring complexity:** RAMI 4.0 is a three-dimensional model that helps to address the problem of Industry 4.0 in a structured manner.
- **Integrating components:** Combines the IT elements and components into a layered model of the lifecycle. This includes aspects such as data privacy and IT security.

RAMI 4.0 as an Architectural Model. Previously discussed as a three-dimensional reference framework for Industry 4.0. Its dimensions include Architecture Layers, the Life Cycle / Value Stream, and Hierarchy Levels. RAMI 4.0 helps to structure and understand the complexity of Industry 4.0 systems and facilitates the integration of components and data [22].

3. Lattice definition in the context of our research

Algebraically, a lattice is an abelian group that spans a vector space; geometrically, it forms a grid. The space exists only at the lattice nodes, meaning that entities

(components, objects, amino acids, molecules, particles) are constrained to these fixed positions [6, 14].

A Walk on a lattice is defined as a sequence of adjacent nodes and serves as a way to encode spatial configurations. In a two-dimensional (2D) square lattice, each node has four nearest neighbors, corresponding to the four cardinal directions: right (R), up (U), left (L), and down (D). In absolute encoding, each step of the walk is represented by one of these direction symbols.

In a three-dimensional (3D) cubic lattice, each node has six nearest neighbors, aligned with the three spatial axes. The absolute directions are: right (R), left (L), up (U), down (D), forward (F), and backward (B). Thus, in 3D, each step in the walk is encoded using one of these six direction symbols. Figure 1a and 1b show the representation of a 2D square lattice and 3D cubic lattice, respectively [20].

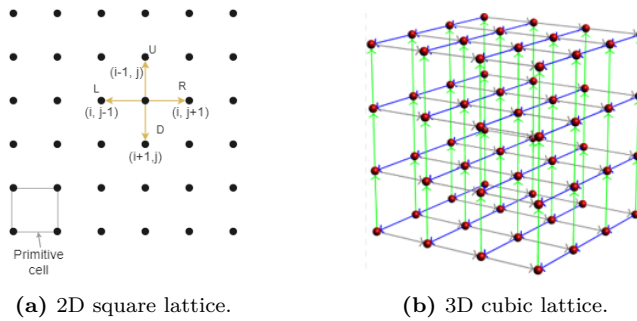


Figure 1. Lattices visual overview.

4. A novel customised architecture of RAMI 4.0

An innovative approach is introduced in this paper, specifically for analysing and interpreting information within IoT environments. This approach utilises a three-dimensional cubic lattice model.

While existing sources address IoT data analysis, varied architectures (including the 3D RAMI 4.0 model), and advanced communication technologies like 5G, this paper introduces a novel, innovative methodology: a three-dimensional cubic lattice model for the analysis and interpretation of data in 5G-based IoT environments.

IoT offers the connectivity and data necessary for operational execution, while RAMI 4.0 provides the architectural framework to organise and contextualise these capabilities by linking physical assets to the digital IoT ecosystem.

4.1. Formal mapping of cRAMI 4.0 from a cubic lattice structure to Petri nets

The RAMI 4.0 provides a multidimensional framework for modeling industrial systems, typically structured along three conceptual axes: hierarchy levels, product

life cycle, and IT/OT layers. In this work, we propose a customized interpretation of these axes of the RAMI 4.0, that we called cRAMI 4.0, more directly aligned with practical aspects of IoT system design and management. Specifically, we define the following mapping:

- **X-axis (horizontal axis to the left) – Detection Level Functions:** This axis represents the range of sensing and detection functionalities in the IoT system, from basic physical sensors to higher-level data acquisition and pre-processing modules. Each step along this axis corresponds to increasing levels of functional complexity and semantic interpretation of raw data.
- **Y-axis (horizontal axis to the right) – Analysis and Maintenance Processes:** This axis models the procedural flow from data collection to fault diagnosis, maintenance planning, and system reconfiguration. It reflects the temporal and logical sequence of actions applied to maintain or enhance system performance.
- **Z-axis (vertical axis) – Access Control Levels:** This axis captures the vertical segmentation of system access permissions, from low-level device access (e.g., automated processes or field operators) to higher-level administrative or supervisory roles. It enables the modeling of security policies and role-based data accessibility across the system.

By structuring the system along these three axes, the cubic lattice model allows for a granular representation of each component and its interactions.

To simulate the behavior of the proposed cRAMI 4.0 architecture, we formally map each lattice node (x_i, y_j, z_k) to a Petri net **place** representing a system state. Each node in the lattice denotes a specific configuration defined by a triplet: (x_i, y_j, z_k) , where x_i a specific detection-level function (e.g., sensor activation, data acquisition), y_j to a stage in the analysis or maintenance process (e.g., fault diagnosis, reconfiguration), and z_k to an access control level (e.g., field operator, supervisor, administrator)

This mapping provides a powerful abstraction for identifying, analyzing, and optimizing information flows across the IoT system. It enables a spatial understanding of data paths, process dependencies, and security constraints, offering a comprehensive perspective that bridges functionality, operations, and governance within Industry 4.0 environments.

We model a cloud-connected smart manufacturing system comprising: a **Perception Layer**: temperature and vibration sensors on industrial machines; a **Network Layer**: edge gateways and routers; and an **Application Layer**: cloud-based analytics and decision support. Each layer is encoded in the lattice and mapped to Petri net zones. The system includes: 18 places (P) representing discrete states; 12 transitions (T) modeling actions and data movement and 3 zones: IoT, connection interface, and cloud. **Transitions** in the Petri net represent the data flow between adjacent detection levels, the procedural advancement in analysis/maintenance and the role-based access control enforcement.

Figure 2 presents the main architectural model presented above. In the lattice representation, the X-axis corresponds to detection level functions (left to right), the Y-axis represents the progression of analysis and maintenance processes, and the Z-axis extends in depth, capturing the hierarchical access control levels.

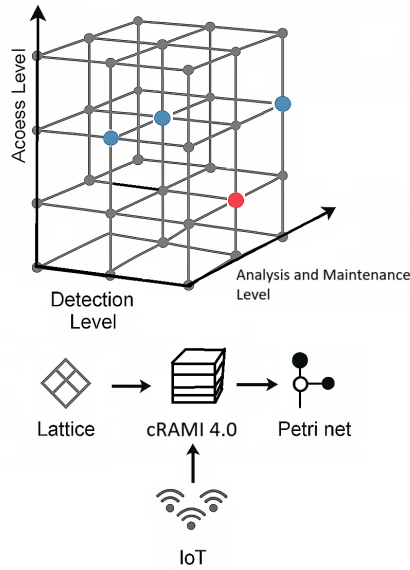


Figure 2. The proposed cRAMI 4.0 Architecture.

The current study's contributions are based on three main goals related to IoT and Cloud, which are also essential elements of RAMI 4.0 [3]:

1. The fundamental study of Petri nets for analysis and validation within discrete systems.
2. Application of Petri nets for modeling and analyzing discrete-event systems, particularly within the detection and analysis layers of IoT architectures.
3. Compare deterministic vs. stochastic behavior in process transitions. Validation methods and results obtained from the analysis of the subject model, which are used to reorganize and reevaluate the system and increase its flexibility. Stochastic modeling introduces variability in transition delays.

This hybrid architecture combines formal modelling (Petri nets), distributed computing (cloud), cyber-physical sensing (IoT) and structured analysis (3D lattices). It exemplifies the integration of computational models [2, 5] with domain-specific goals, such as those in smart manufacturing, health, or molecular biology.

4.2. Simulations using Petri nets

To model and simulate the proposed architecture, we adopt Petri nets (the simulations were performed in Visual Object Net++ application software [7]), which are relatively easy to use, cost-effective, provide real-time insight, and support dynamic intervention without deviating from the systems original purpose.

Table 1. Simulation parameters and variables.

Parameter	Description	Value Range
Sensor activation rate	Frequency of data generation	1–10 Hz
Transition delay	Time between state changes	0.1–2.0 s
Access level	Role-based permission tier	1 (low) to 3 (high)
Token count	Number of active processes	0–100
Fault injection rate	Probability of error occurrence	0–0.2
Data throughput	Volume of data per unit time	10–1000 KB/s

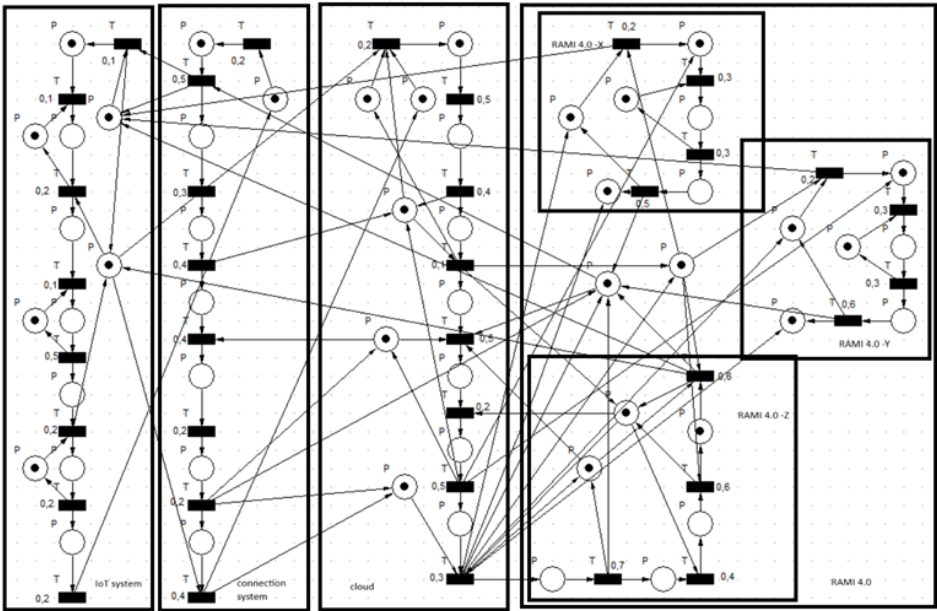


Figure 3. General Petri nets model architecture.

Figure 3 illustrates the key stages of an ideal cloud manufacturing system built based on the cRAMI architectural model Petri net structure: P = place (state), T = transition (action), mXX = monitored metric. All elements used in the simulation are discrete, which can be difficult to achieve in practice. The modeled system is

schematic and contains the basic components of IoT, cloud, cRAMI 4.0, and their interconnections. It is a generic system designed to capture the main structural elements.

During the simulations, intense activity is observed throughout the requested time window in the IoT zone. In the Figures 4–6, flow variations across IoT, connection, and cloud zones takes place. In the Cloud zone, responses occur only at required levels. The diagram displays three fundamental levels: initial, intermediate (monitoring), and final stages of the process. This branch serves as a bridge between the IoT and the rest of the system, ensuring data security and seamless information transfer throughout the entire structure (see Figure 4).

Figure 5 presents both the variation in input data flow and the dynamics of information handled within the cloud system, with real-time accessibility from both RAMI 4.0 and IoT. Figure 6 shows variation in information flow caused by input data and the systems processing capacity.

Figures 7, 8 and Figure 9 illustrate the ordered information dynamics along each cRAMI axis ($X = \text{detection}$, $Y = \text{analysis}$, $Z = \text{access control}$). Variations are caused by either the high volume of data or complex data structures in the system. As seen in Figure 9, large variations can arise due to an error within the analysis process. Since the system is secure, other components are not severely impacted. The error may stem from sudden data fluctuation, hardware malfunction, or even human error in the case of incorrect data routing.

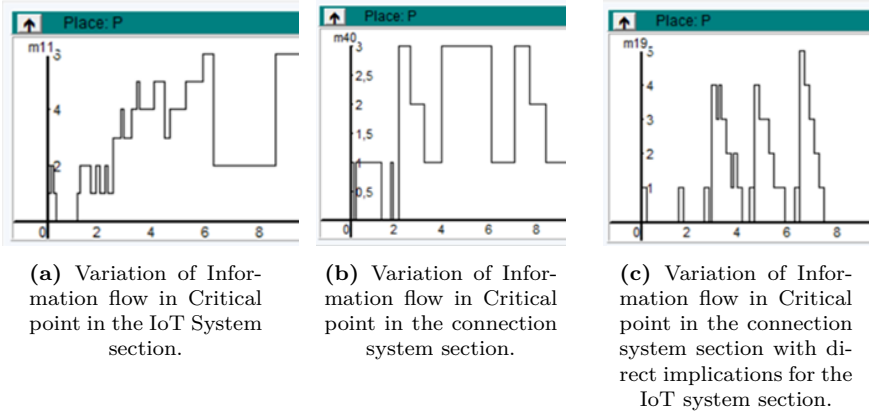


Figure 4. Information flow variation between the IoT system and the connection system.

5. Discussion

This complex system benefits from a well-developed and relatively accessible IoT component, crucial for monitoring and communication. In our model, the IoT

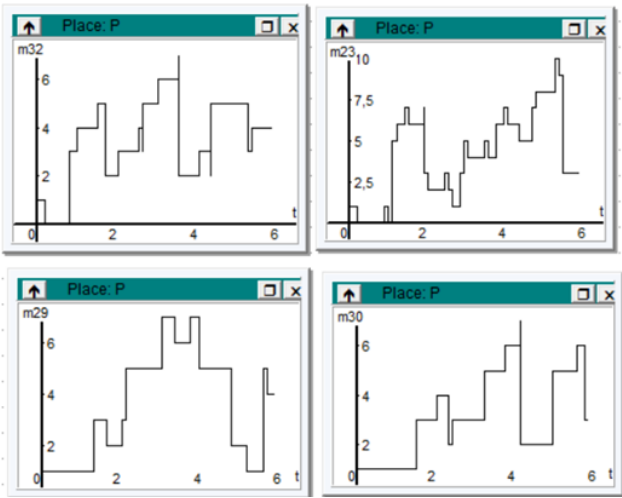


Figure 5. Information flow variation within the cloud system.

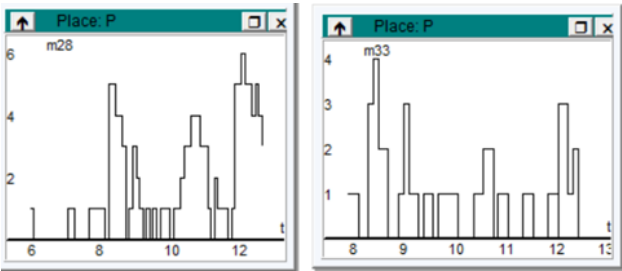


Figure 6. Information flow between RAMI 4.0 and the system connected to the cloud.

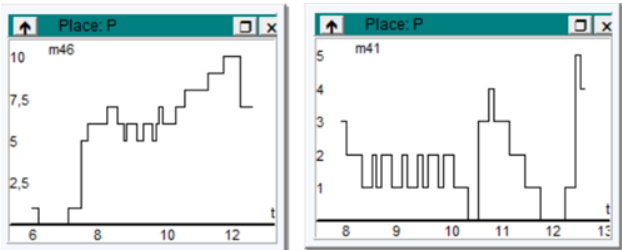


Figure 7. Information flow along RAMI 4.0 - X direction.

system is linked to the cloud via a dedicated connection component that ensures data security and facilitates efficient information flow. cRAMI 4.0 decomposes into its three core axes, each mapped to its specific structure, with no overlap in data,

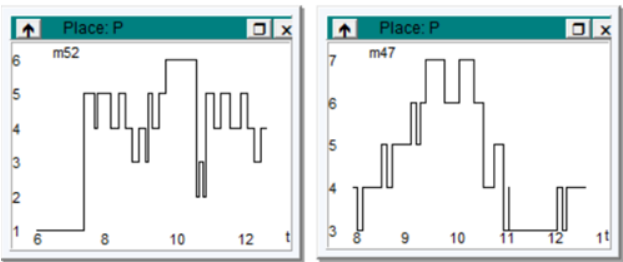


Figure 8. Information flow along RAMI 4.0 - Y direction.

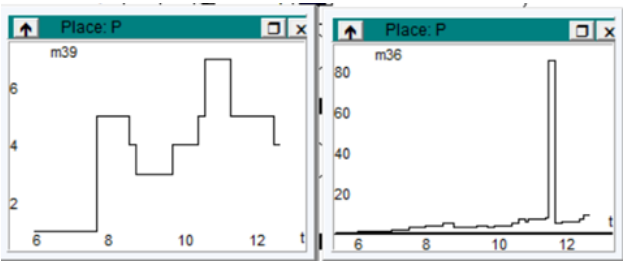


Figure 9. Information flow along RAMI 4.0 - Z direction.

thanks to the interconnectivity of the cube surface. Functional and communication mappings are preserved for every axis.

Challenges related to this approach include the need for novel data models and coordination mechanisms between cloud devices, cloud servers, and artificial intelligence systems. Also, important integration of advanced data quality assessment methods like those presented in [12]. Consequently, the IoT architecture for cloud-based devices has been conceived as a modular framework that supports low-power wearable and implantable devices, compatible with others using similar data formats and capable of wireless interaction.

Limitations and challenges. Existing models and systems still face significant limitations that must be addressed to ensure the effective use of cloud devices in dedicated systems. These include issues such as energy consumption, optimal data scheduling, low-latency models, privacy, joint offloading, limited data availability, data security, system downscaling, and overhead costs. To overcome these, further exploration of new architectures, development of optimised training techniques, interpretable models and architectures for resource-constrained environments are needed. Ethical and concerns also require attention to prevent bias and discrimination, especially in sensitive application areas.

6. Conclusions

This paper proposed a three-dimensional cubic lattice model for structuring and optimizing information flows in IoT systems, based on a customized interpretation of the RAMI 4.0 framework, called cRAMI 4.0. This paper proposes a three-dimensional cubic lattice model for structuring and optimizing information flows in IoT systems, based on a customized interpretation of the Reference Architectural Model for Industry 4.0 (RAMI 4.0), referred to as **customized RAMI 4.0 (cRAMI 4.0)**. The model defines three orthogonal axes, detection level functions, analysis and maintenance processes, and access control levels, allowing a spatial representation of the system's operational, procedural, and security dimensions. The specific lattice-based representation enables a clear visualization of interactions, supports granular system analysis, and facilitates optimization of data flow and security mechanisms. Systems based on cRAMI 4.0 can be utilized, offering grouped benefits, real-time responsiveness, and low data and energy consumption. It supports the development of multiple applications across different sectors, tailored to user-specific requirements. The knowledge obtained is more structured, and more relevant for decision-making in developing Industry 4.0 developments.

To validate this approach, simulations were conducted using Petri nets in Visual Object Net++, enabling graphical representation and analysis of information flow dynamics across all three axes. The results confirm the models suitability for identifying critical paths, analyzing interdependencies, and supporting optimization strategies in complex IoT architectures. Future work will focus on extending the model with dynamic reconfiguration capabilities and integrating real-time system data.

Acknowledgements. This research received financial support from ‘1 Decembrie 1918’ University of Alba Iulia, Romania, through Order no. 3259 of 13th February 2025. The authors thank the Research Group on Artificial Intelligence and Data Science for Healthcare Innovation (REFLECTION) and the Research Center on Artificial Intelligence, Data Science, and Smart Engineering (ARTEMIS), of the George Emil Palade University of Medicine, Pharmacy, Science and Technology of Targu Mures, Romania, for support of research infrastructure. The WG6 Benchmarking Workgroup within CA22137 COST Action, Randomized Optimization Algorithms Research Network (ROAR-NET), concerned with identifying benchmarking issues.

References

- [1] R. BERGENTHUM, J. KOVÁŘ: *Synthesizing Petri Nets from Labelled Petri Nets*, in: Application and Theory of Petri Nets and Concurrency, ed. by E. AMPARORE, L. MIKULSKI, Cham: Springer Nature Switzerland, 2025, pp. 63–85, ISBN: 978-3-031-94634-9.

- [2] L. L. BOCA, E. M. CIORTEA, C. BOGHEAN, A. BEGOV-UNGUR, F. BOGHEAN, V. T. DĂDĂRLAT: *An IoT System Proposed for Higher Education: Approaches and Challenges in Economics, Computational Linguistics, and Engineering*, Sensors 23.14 (2023), ISSN: 1424-8220, DOI: [10.3390/s23146272](https://doi.org/10.3390/s23146272).
- [3] E. M. CIORTEA: *Cloud Manufacturing - The Connection Between RAMI 4.0 and IoT*, in: IOP Conference Series: Materials Science and Engineering, vol. 916, IOP Publishing, 2020, p. 012019, DOI: [10.1088/1757-899X/916/1/012019](https://doi.org/10.1088/1757-899X/916/1/012019).
- [4] A. W. COLOMBO, S. KARNOUSKOS, T. BANGEMANN: *Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges*, in: Proceedings of the IEEE, vol. 104, 5, 2017, pp. 1107–1123, DOI: [10.1109/JPROC.2016.2554518](https://doi.org/10.1109/JPROC.2016.2554518).
- [5] D. CRISTEA: *Hybrid Combinatorial Problems Used for Multimodal Optimisation*, in: 2024 IEEE 24th International Conference on Bioinformatics and Bioengineering (BIBE), 2024, pp. 1–8.
- [6] K. A. DILL: *Theory for the folding and stability of globular proteins*, Biochemistry 24 (1985), p. 1501, DOI: [10.1021/bi00327a032](https://doi.org/10.1021/bi00327a032).
- [7] R. DRATH: *Visual Object Net++: Petri-Net CAD/CAE Tool Supporting Hybrid Petri Nets*, Software (Version 2.7a) available online, URL: https://www.r-drath.de/Home/Visual_Object_Net++.html.
- [8] B. GUERROUDJ, A. SIAM: *The Importance of Semantic Interoperability in The Internet of Things (IoT)*, in: Joint Proceedings of the Second International Workshop on Semantic Reasoning and Representation in IoT (SWIoT 2023) and the Third International Workshop on Multilingual Semantic Web (MSW 2023), Zaragoza, Nov. 2023.
- [9] E. HERNÁNDEZ, P. SENNA, D. SILVA, R. REBELO, A. BARROS, C. TOSCANO: *Implementing RAMI4.0 in Production – a Multi-Case Study*, in: Progress in Digital and Physical Manufacturing, 2020, DOI: [10.1007/978-3-030-29041-2_6](https://doi.org/10.1007/978-3-030-29041-2_6).
- [10] L. B. IANTOVICS: *Black-Box-Based Mathematical Modelling of Machine Intelligence Measuring*, Mathematics 9.6 (2021), ISSN: 2227-7390, DOI: [10.3390/math9060681](https://doi.org/10.3390/math9060681).
- [11] L. B. IANTOVICS, F. EMMERT-STREIB, S. ARIK: *MetrIntMeas a novel metric for measuring the intelligence of a swarm of cooperating agents*, Cognitive Systems Research 45 (2017), pp. 17–29, ISSN: 1389-0417, DOI: [10.1016/j.cogsys.2017.04.006](https://doi.org/10.1016/j.cogsys.2017.04.006).
- [12] L. B. IANTOVICS, C. ENĂCHESCU: *Method for Data Quality Assessment of Synthetic Industrial Data*, Sensors 22.4 (2022), ISSN: 1424-8220, DOI: [10.3390/s22041608](https://doi.org/10.3390/s22041608).
- [13] S. JAVED ET AL.: *Visualisation Approach for RAMI 4.0 Value Chain Analysis*, Industrial Electronics Society (2024), DOI: [10.1109/OJIES.2024.3520410](https://doi.org/10.1109/OJIES.2024.3520410).
- [14] K. F. LAU, K. A. DILL: *A lattice statistical mechanics model of the conformational and sequence spaces of proteins*, Macromolecules 22.10 (1989), pp. 3986–3997, DOI: [10.1021/ma00200a030](https://doi.org/10.1021/ma00200a030).
- [15] R. S. MENDONCA, R. L. P. MEDEIROS, L. E. S. E. SILVA, R. G. G. SILVA, L. G. S. SANTOS, V. F. DE LUCENA: *Enabling Technologies of Industry 4.0 for the Modernization of an Industrial Process*, Processes 13.8 (2025), ISSN: 2227-9717, DOI: [10.3390/pr13082488](https://doi.org/10.3390/pr13082488).
- [16] T. MURATA: *Petri nets: Properties, analysis and applications*, Proceedings of the IEEE 77.4 (1989), pp. 541–580, DOI: [10.1109/5.24143](https://doi.org/10.1109/5.24143).
- [17] V. A. OKHUESE, M. I. ALI: *Healthcare Internet of Things (IoT): A Survey of State-of-the-art Methods and Approaches*, TechRxiv (2024), CC BY 4.0 License, DOI: [10.36227/techrxiv.24080559](https://doi.org/10.36227/techrxiv.24080559).
- [18] A. SHIRBAZO ET AL.: *A Guideline for the Standardization of Smart Manufacturing and the Role of RAMI 4.0 in Digitising the Industrial Sector*, IEEE Internet of Things Journal 12.12 (2025), pp. 19090–19118, DOI: [10.1109/JIOT.2025.3559929](https://doi.org/10.1109/JIOT.2025.3559929).
- [19] A. J. SILVA, P. CORTEZ, C. PEREIRA, A. PILASTRI: *Business Analytics in Industry 4.0: A Systematic Review*, Expert Systems 38.3 (2021), DOI: [10.1111/exsy.1274](https://doi.org/10.1111/exsy.1274).

- [20] I. SIMA, D.-M. CRISTEA: *Record-to-Record Travel Algorithm for Biomolecules Structure Prediction*, in: Computational Science and Its Applications – ICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–16, 2021, Proceedings, Part I, Berlin, Heidelberg: Springer-Verlag, 2021, pp. 449–464, ISBN: 978-3-030-86652-5, DOI: [10.1007/978-3-030-86653-2_33](https://doi.org/10.1007/978-3-030-86653-2_33).
- [21] D. THIELE, F. WITTE, A. FAY: *Using Petri nets for modeling and verification of industrial IoT applications within RAMI 4.0*, in: 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), 2019, pp. 1448–1453, DOI: [10.1109/INDIN41052.2019.8972287](https://doi.org/10.1109/INDIN41052.2019.8972287).
- [22] M. ZAHEER, R. FAIZ, S. ABBAS: *Industrial Challenges of Security Threats upon Security Related IoT Components in RAMI 4.0*, Journal of Computer Engineering and Information Technology 10.10 (2021), p. 291.

Betweenness-driven overlapping label propagation community detection*

Sylvvert Prian Tahalea^{ab}, Miklós Krész^{cd}

^aDoctoral School of Computer Science, University of Szeged, Hungary
sylvvert@inf.u-szeged.hu

^bDepartment of Informatics, Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia

^cDepartment of Applied Informatics, University of Szeged, Bolodgasszony sgt. 6, H-6725 Szeged, Hungary

^dInnoRenew CoE, UP IAM and UP FAMNIT, University of Primorska, Titov trg 4, 6000 Koper, Slovenia
miklos.kresz@innorenew.eu

Abstract. Community detection holds significant value in discovering hidden structures in complex networks. In this paper, we propose a betweenness-driven community detection based on the label propagation algorithm. First, at the multiple labels' assignment phase, we detect communities using label propagation and apply labels for the nodes using the betweenness and degree centrality as references. Second, we refine the modularity and stability using several configurations, such as global modularity and stability pruning, to avoid nodes that have not changed for several iterations. This algorithm was tested with the most common datasets, such as Zachary's Karate Club Network, Polbooks, Football, and 12 LFR synthetic datasets, which resulted in improved scores on modularity, overlapping normalised mutual information, omega index, generalized F1-score, and extended various pieces of information.

Keywords: network, graph, overlapping, community detection, label propagation algorithm

*The research was supported by the BioLOG project: the second author is grateful for the support of the National Centre of Science (NCN) through grant DEC-2020/39/I/HS4/03533, the Slovenian Research and Innovation Agency (ARIS) through grant N1-0223 and the Austrian Science Fund (FWF) through grant I 5443-N. This work is supported by the ARIS research program P1-0404 and by the research program CogniCom (0013103) at the University of Primorska.

1. Introduction

Network analysis is implemented and used in various disciplines to represent their complex interactions, such as social sciences, computer sciences, biology, and materials science. Networks consisting of nodes with edges connecting them encode information through their structural characteristics; one of the most popular research areas is the study of community structures. While it has no formal definition, a community can be considered a densely connected subgraph of a network, which means that nodes within the community have strong relationships but are sparsely connected to the rest of the network. Generally, there are two types of communities: non-overlapping or disjoint communities and overlapping communities. The non-overlapping community detection separated the nodes based on their membership, where each node only belongs to exactly one community. Meanwhile, the overlapping communities consider the nodes that belong to one or more communities.

Community detection is considered an important task because it can uncover the hidden structure of a complex network. Most community detection algorithms have been developed to solve the non-overlapping community detection problem, but some algorithms work well to solve both non-overlapping and overlapping community detection problems [4]. Many approaches have been proposed to solve the community detection problems, such as clique percolation approach [23], label propagation [25], non-negative matrix factorization (NMF) [28], fuzzy set theory [8], evolutionary algorithms [24], and even the statistical models [9]. The Newman-Girvan modularity measurements have become one of the most popular methods to measure the density of communities within the network [21], since they provide an objective way to evaluate the communities' quality. This measurement indicates that nodes are more closely connected to their community compared to other nodes in the network. A modularity score near zero indicates that there is no real community structure, while a score near one means the communities are dense and well-structured. The label propagation algorithm (LPA) approach has received a lot of interest because of its simplicity and scalability [7]. As an extension of its original non-overlapping version [25], it consists of simple steps for the overlapping community detection [8], such as 1) label every node with its unique label; 2) label the current node based on its neighbours' labels; 3) propagate for all the nodes in the network; 4) compare the label of each node in the current iteration with the previous iteration; and finally 5) labels indicate the communities for each node (a node may have multiple labels). The weighting system is usually applied in stage 3. The iteration is terminated at stage 4 if convergence occurs; otherwise, another iteration is executed.

There are several notable overlapping community algorithms based on label propagation. In 2010, Gregory [8] designed a specific algorithm called COPRA (Community Overlap Propagation Algorithm) to allow nodes to hold multiple labels simultaneously using the belonging coefficient as a degree of membership. Later, the dynamic process where each node acts as a speaker and a listener was

introduced as SLPA [29], which identified both the number and the strength of each node's community affiliation. Another approach, which combined with LPA is a local spectral method called LEMON [15]: a sparse vector is obtained by minimising the ℓ_1 norm over a local spectral subspace with seed constraints, which ensures the seed nodes to be included and highlights additional nodes to include in the community.

In 2021, Attal et. al. [1] introduced a method to find overlapping community detection using pre-computed disjoint communities. This algorithm, leveraging density and clustering coefficient as belonging function, compares to closeness and betweenness centrality as average node measures, defining a node's memberships. These results show that communities with high density and clustering coefficient performed better than closeness and betweenness centrality measures. In the same year, Li and Sun [14] introduced a combination of local expansion and label propagation (LELP), which uses local expansion to generate some immature communities, prunes the network, and uses LPA to obtain a stable network.

In 2022, density based-label propagation algorithm (D-LPA) [31] was developed, combining the density peak clustering with traditional LPA to improve the stability and accuracy of community assignments. The vector-label propagation algorithm (vLPA) was also introduced in this year [3], where gradient descent was utilised to improve the modularity. This approach retains weak structural information, but obtains better performance when the community structure is weak. The influence-based COPRA approach introduced as INF-COPRA [30], is an algorithm that ranks the influence of nodes and labels, thereby improving the extended modularity (EQ) and normalised mutual information. One of the latest expansions of LPA was the degree and betweenness-based label propagation (DBLPA) [22], which combines the degree and betweenness centrality to provide the core nodes in layer-by-layer LPA [34].

Label propagation algorithms have a fast runtime but have strong randomness and weak robustness causing difficulties in obtaining effective community detection results. The expansion of the label propagation algorithms conducted with several approaches, such as the multistep greedy, to increase modularity, but sacrifices the fast-running time [17]. There is also a kernel label approach proposed to reduce the complexity, but at the same time improve the randomness of the algorithm [16]. Meanwhile, others proposed the accelerated modularity gain by analysing Newman's modularity function [33].

Overlapping community detection is important because real-world networks rarely consist of cleanly separated groups. Detecting overlapping communities can lead to the nature of multiple roles of nodes, provide more accurate prediction of network behaviour, capture phenomena such as redundancy and robustness in complex systems, and define the characteristics that strengthen the interpretability and predictive utility of community detection methods. Modularity is one of the most widely accepted quality functions in community detection because it measures how well a given community separates dense intra-community connections from sparse inter-community connections. However, overlapping communities often underes-

timates the quality of the community structure because nodes can contribute to multiple groups simultaneously. If the overlapping assignments are not refined, the modularity may remain low, indicating communities are not cohesive or not well-defined. Improving modularity in overlapping community detection ensures the structural density and avoids the risk of generating fragmented communities.

Betweenness centrality quantifies the shortest paths between others, showing a strong indicator of a boundary in the networks. In overlapping community detection, betweenness nodes are precisely where community memberships are likely to overlap since they act as the bridge across different groups of nodes in the network. This betweenness-driven approach leverages the inverse betweenness, causing these nodes to retain multiple memberships in the overlap assignment phase. This approach utilises the balance between degree centrality capturing the local influences and betweenness centrality, which highlights global structure.

This paper proposes a novel overlapping community detection algorithm utilising the fast propagation through the network of LPA combined with the nature of the betweenness centrality score of a node. This algorithm is split into two phases: multi-assignment label propagation and modularity refinement. In the first phase, the algorithm will quickly build an overlapping community structure using LPA with betweenness as its voting mechanism to decide the memberships of each node. This allows strong local leaders to influence community growth while boundary nodes exert more measured influence. In the second phase, modularity refinement is executed using Newman’s modularity [21] with temporal projections to refine the modularity of the communities within the network. Finally, overlapping communities are defined by assigning the nodes to all communities where their membership strength exceeds the threshold. In this proposed algorithm, several adaptation parameters are used to maximise the modularity, such as top- k filtering to make sure that the nodes can hold k maximum labels as their possible communities, and minimum gain as a threshold when maximising the modularity.

The main contributions of this paper are as follows:

1. Balancing high-speed and high-quality communities, addressing the weaknesses of the classic label propagation algorithm.
2. Flexible filtering strategies (such as Top-K filtering, minimum gain threshold, and stability pruning) which lead to an increase in the quality of communities produced.
3. Utilising the nature of betweenness nodes to define the overlapping nodes.

The remainder of the paper is organised as follows. Section 2 explains the methodology, data, and evaluation techniques used in this study. Section 3 presents the main results, with tables highlighting the main findings. Section 4 concludes the contribution, takeaways, and future research.

2. Methodology

The Betweenness-Driven Label Propagation Algorithm (BD-LPA) is an overlapping community detection method that combines local label propagation with a global modularity-based refinement. The high-betweenness nodes often sit at community boundaries and influence how strongly a node is pulled by its neighbours' labels. By seeding each node with a unique label and weighting those labels according to both how often they appear in the neighbourhood and how central their owners are, BD-LPA builds a soft membership vector for every node. This vector encodes the node's membership towards multiple communities, allowing for overlapping structures. The algorithm proceeds in two main phases—first, a fast, distributed voting scheme that spreads labels with influence proportional to neighbour degree and inverse to neighbour betweenness, and then a slower, global refinement that uses modularity gains to reinforce coherent groupings. This proposed method is split into two phases, namely the multi-assignment label propagation phase and the global modularity refinement phase.

2.1. Degree and betweenness centrality

Degree centrality measures the number of edges connected to a node [5]. This centrality shows the position of a node in the networks based on its connections and can be measured as follows. For a simple undirected graph $G = (V, E)$, the degree centrality C_d of node v is

$$C_D(v) = \deg(v),$$

where $\deg(v)$ is the degree of node v .

While degree centrality measures the node's centrality using the direct connection to the node, betweenness centrality measures the node's centrality based on how often the node lies on the shortest path between other nodes [6]. Betweenness centrality C_b of a node v can be measured as follows. Let σ_{xy} be the number of shortest path between nodes x and y , and $\sigma_{xy}(v)$ the number of paths that pass through node v , with $v \neq x \neq y$. The betweenness centrality of v is

$$C_B(v) = \sum_{v \neq x \neq y} \frac{\sigma_{xy}(v)}{\sigma_{xy}}.$$

The nature of a node with a high betweenness value is becoming a hub for the network, meaning that they are node with a high possibility of having multiple community memberships. On the contrary, nodes with low betweenness are more likely to have single community membership but not necessarily rely only on their betweenness centrality value. Thus, in this study, we propose to combine the use of degree and betweenness centrality as the community membership voting mechanism.

2.2. Multi-assignment label propagation

In Phase 1, each node maintains a weight vector over candidate community labels. In each iteration, nodes are visited in random order and vote for labels: each neighbour contributes to the vote for its own dominant label in proportion to its degree divided by one plus its betweenness centrality. The candidate labels with the highest votes are then added to the node's weight vector with a fixed gain factor, and the vector is renormalised as probabilities, retaining only the top L labels. Over multiple fast iterations, this process diffuses label influence through the graph, enabling nodes at the fringes of communities to accumulate membership probabilities in several nearby groups. Since the voting weight uses degree and penalises high-betweenness nodes, labels spread more readily within densely connected regions while respecting bottlenecks. The node membership voting mechanism can be computed as follows.

$$vote[\ell_u^*] += \frac{\deg(u)}{1 + \text{btw}(u)},$$

where ℓ_u^* is the dominant label of neighbor u , $\deg(u)$ is the degree of node u , $\text{btw}(u)$ is the betweenness centrality of node u , and $vote[\ell_u^*]$ is the cumulative vote weight for label received by node u from its neighbors. The algorithm for phase 1 is presented in Algorithm 1.

2.3. Modularity refinement

In Phase 2, modularity refinement measurement utilises Newmann's modularity with temporary projection or hard mapping[26, 27]. This shifts the measurement from local diffusion to global optimisation by temporarily hard-assigning each node to its highest-weight label and computing the resulting modularity. It then attempts to improve modularity by considering, for each node in random order, re-assigning it to one of its neighbours' labels if it results in a modularity gain above a small threshold. Whenever a better community label is found, the node's weight vector is reset to that single label before moving on. Iterations continue until no single-node swap can further increase modularity. Crucially, these hard-assignment trials only guide the search; at the end of refinement, the algorithm reverts to the soft weight vectors and applies thresholds to produce overlapping communities, preserving multi-membership while ensuring that each switch meaningfully boosts global cohesion. The algorithm for phase 2 is presented in Algorithm 2, and the modularity measurement is conducted as follows.

1. After phase-1, each node v has membership weight $w_v(\ell)$
2. Temporay projection (hard mapping) performed as $c_i^* = \arg \max_{\ell} w_i(\ell)$
3. Compute the modularity using Newman's modularity [21] for the temporary hard mapping as follows.

$$Q^{proj} = \frac{1}{2m} \sum [A_{ij} \frac{k_i k_j}{2m}] \delta(c_i^*, c_j^*), \quad (2.1)$$

Algorithm 1 Phase 1: Multi-Assignment Label Propagation

Require: $G = (V, E)$ ▷ undirected graph
Require: T_f ▷ fast propagation iterations
Require: k ▷ top- k votes per node
Require: $L_{\max}$ ▷ max labels per node
Require: η ▷ vote gain factor
Ensure: $\{w_v\}$ ▷ soft label weights

```

1: // Initialisation
2: for each  $v \in V$  do
3:   assign unique label  $\ell_v$ 
4:    $w_v \leftarrow \{\ell_v \mapsto 1.0\}$ 
5: end for
6: // Propagation
7: for  $t = 1$  to  $T_f$  do
8:   SHUFFLE( $V$ )
9:   for each  $v \in V$  do
10:     $vote \leftarrow \{\}$ 
11:    for each  $u \in N(v)$  do
12:       $\ell_u^* \leftarrow \arg \max_{\ell} w_u[\ell]$ 
13:       $vote[\ell_u^*] += \frac{\deg(u)}{1 + \text{btw}(u)}$ 
14:    end for
15:     $T \leftarrow$  top- $k$  labels by  $vote$ 
16:    for each  $\ell \in T$  do
17:       $w_v[\ell] += \eta \text{ vote}[\ell]$ 
18:    end for
19:     $w_v \leftarrow \text{TruncatedSoftmax}(w_v, L_{\max})$ 
20:  end for
21: end for

```

where A_{ij} is actual adjacency between node i and j ; k_i, k_j are the degrees of node i and j ; m is the total number of edges; and $\delta(c_i^*, c_j^*)$ shows the position of node i and j on their temporary mapping, 1 means both nodes are in the same community, and 0 otherwise.

Here is an example of the approach.

1. Graph: G with $V = \{A, B, C, D\}$, a 4-cycle, $m = 4$
2. Communities: $C_1 = \{A, B, C\}, C_2 = \{A, C, D\}$
3. Assumed that memberships after phase-1:
 - $A : C_1 : 0.7, C_2 : 0.3$
 - $B : C_1 : 0.6, C_2 : 0.4$
 - $C : C_1 : 0.2, C_2 : 0.8$
 - $D : C_1 : 0.1, C_2 : 0.9$

4. Temporal projection:

- $A \mapsto C_1$
- $B \mapsto C_1$
- $C \mapsto C_2$
- $D \mapsto C_2$
- Projected community $C_1^* = \{A, B, C\}, C_2^* = \{D\}$

5. Using equation (2.1) $Q^{proj} = 0.08$

2.4. Evaluation

This proposed algorithm was evaluated using several standard metrics that are widely adopted, such as modularity, overlapping normalized mutual information (ONMI), omega index, generalized F1-score, and extended variation of information (extended VI) as the proper evaluation for overlapping community detection [7, 11]. Modularity introduced by Newman and Girvan [21] to quantify the internal structure of the community, while ONMI measures the similarity between the results and the ground truth [18] as follows:

$$NMI_{max} = \frac{I(X : Y)}{\max(H(X), H(Y))},$$

where $I(X : Y) = \frac{1}{2}[H(X) - H(X|Y) + H(Y) - H(X|Y)]$ is the mutual information and $H(X|Y) = \sum_{i \in 1, \dots, K_X}$ is the total information.

To evaluate the accuracy, imbalance community, and discrepancy in shared information, the omega index was used [20] as follows.

$$\Omega(C_1, C_2) = \frac{o_u(C_1, C_2) - o_e(C_1, C_2)}{1 - o_e(C_1, C_2)},$$

where $o_u(C_1, C_2)$ is the fraction of pairs that occur in the same number of communities in both communities, and $o_e(C_1, C_2)$ is the expected fraction under random assignment.

The generalized F1-score [32] is also used to measure the best-matching community produced by the algorithm compared with the ground truth, which can be measured as follows.

$$\frac{1}{2} \left(\frac{1}{|C^*|} \sum_{C_i \in C^*} F1(C_i, \hat{C}_{g(i)}) + \frac{1}{|\hat{C}|} \sum_{\hat{C}_i \in \hat{C}} F1(C_{g'(i)}, \hat{C}_i) \right),$$

where the best matching g and g' is defined as $g(i) = \arg \max_j F1(C_i, \hat{C}_j)$ and $g'(i) = \arg \max_j F1(C_j, \hat{C}_i)$.

Algorithm 2 Phase 2: Modularity Refinement

Require: $G = (V, E)$ ▷ undirected graph
Require: $\{w_v\}_{v \in V}$ ▷ weight vectors from Phase 1
Require: T_r ▷ refinement iterations
Require: ε ▷ modularity-gain threshold
Ensure: $\{w_v\}_{v \in V}$ ▷ refined weight vectors

```

1: function HARDMAP( $v$ )
2:   return  $\arg \max_{\ell} w_v[\ell]$ 
3: end function
4: function BUILDCOMMUNITIES
5:    $C \leftarrow \{\}$  ▷ map label→node list
6:   for each  $v \in V$  do
7:      $\ell \leftarrow \text{HARDMAP}(v)$ 
8:     append  $v$  to  $C[\ell]$ 
9:   end for
10:  return list of communities in  $C$ 
11: end function
12:  $Q \leftarrow \text{modularity}(G, \text{BUILDCOMMUNITIES})$ 
13: for  $t \leftarrow 1$  to  $T_r$  do
14:   changed  $\leftarrow$  false; SHUFFLE( $V$ )
15:   for each  $v \in V$  do
16:      $c_0 \leftarrow \text{HARDMAP}(v)$ 
17:     Cand  $\leftarrow \{\text{HARDMAP}(u) \mid u \in N(v)\} \setminus \{c_0\}$ 
18:     best,  $\Delta_{\text{best}} \leftarrow c_0, 0$ 
19:     for each  $\ell \in \text{Cand}$  do
20:       save  $w_v^{\text{old}} \leftarrow w_v$ 
21:        $w_v \leftarrow \{\ell \mapsto 1.0\}$ 
22:        $Q' \leftarrow \text{modularity}(G, \text{BUILDCOMMUNITIES})$ 
23:        $\Delta \leftarrow Q' - Q$ 
24:       if  $\Delta > \varepsilon$  and  $\Delta > \Delta_{\text{best}}$  then
25:         best,  $\Delta_{\text{best}} \leftarrow \ell, \Delta$ 
26:       end if
27:       restore  $w_v \leftarrow w_v^{\text{old}}$ 
28:     end for
29:     if best  $\neq c_0$  then
30:        $w_v \leftarrow \{\text{best} \mapsto 1.0\}$ 
31:        $Q \leftarrow Q + \Delta_{\text{best}}$ 
32:       changed  $\leftarrow$  true
33:     end if
34:   end for
35:   if not changed then
36:     break
37:   end if
38: end for

```

Another evaluation measurement used was the extended VI [19] as follows.

$$VI(X, Y) = H(X|Y) + H(Y|X),$$

where $H(X|Y)$ is the conditional entropy of community X given community Y and vice versa. Conditional entropy measures the certainty of a community assignment given knowledge of the other communities. It captures the average amount of extra information to describe the communities. A lower conditional entropy implies that knowing one community almost fully explains the other (strong agreement), while a higher value means the communities disagree more, reflecting greater divergence in how overlaps are assigned.

The datasets used in this evaluation are Zachary’s Karate Club Network [5], Football [5], Polbooks [10], and synthetic datasets generated using the LFR framework [12].

3. Experimental results and analysis

The proposed algorithms were benchmarked using four standard metrics, such as modularity, ONMI, Omega Index, Generalized F1-score and Extended VI, over the four datasets mentioned before. The results were compared to three overlapping community detection algorithms such as COPRA [8], SLPA [29], and Motif-LPA [13], to evaluate their performance to the LPA-based algorithm.

- **COPRA:** Expansion of traditional LPA using membership coefficients to allow nodes to belong to multiple communities.
- **SLPA:** Dynamic model of LPA where nodes act as speakers and listeners, allowing nodes to remember the community label and become its member.
- **Motif-LPA:** Introducing a motif-based approach (recurring structural patterns) and capturing high-order connectivity to allow nodes to become members of more than one community

3.1. Datasets

Three real-world datasets are commonly used in community detection benchmarking, utilised in this research. The Zachary’s Karate Club network describes relationships between club members, which are divided into two communities. The Football dataset was created from the American College Football League, where nodes represent the football team and edges represent the game played between them. The Polbooks network was created based on the interaction between readers of American politics books. Alongside the real-world datasets, we generated 12 synthetic datasets using the LFR model [12] with different settings. The LFR model is widely adopted in community detection research because it produces networks with realistic structural features (heterogeneous degree distributions and community sizes) while allowing precise control over the embedded community structure.

Unlike real-world networks, where communities are inferred from contextual meaning, in LFR networks, communities are explicitly defined by the generation process, making them suitable for controlled benchmarking. Each synthetic dataset

in Table 1 is characterised by: n (Nodes): the size and connectivity of the generated network; c (Communities): the number of planted communities generated by the model; k (Average degree): the average number of edges per node; μ (Mixing parameter): the proportion of edges that connect a node to nodes outside its community. A low μ (close to 0) indicates well-separated communities, while a higher μ implies stronger inter-community mixing and weaker community structure; on (Overlapping nodes): the number of nodes that belong to more than one community; om (Overlapping memberships): the number of communities each overlapping node belongs to. For example, $om = 2$ indicates that overlapping nodes belong to exactly two communities, while $om = 3$ allows nodes to be shared across three communities.

The parameterisation across the LFR datasets was chosen to cover different scenarios of network size from 1000 to 10000 nodes, degree distributions, mixing levels, and varying degrees of community overlap. This diversity ensures the algorithm's robustness under a wide range of structural complexities and community structures. Table 1 provides the summary of the datasets, where c is the number of communities, k is the average degree, μ is the mixing parameter, on is overlapping nodes, and om is overlapping memberships.

Table 1. Comparative metadata of benchmark datasets.

Dataset	Nodes	Edges	c	k	μ	on	om
Karate	34	78	2	-	-	-	-
Football	115	613	3	-	-	-	-
Polbooks	105	441	12	-	-	-	-
LFR1	1000	10455	44	20.91	0.10	100	0
LFR2	1000	12555	35	25.11	0.30	100	2
LFR3	1000	6003	27	12.01	0.15	100	3
LFR4	2000	24555	40	24.55	0.20	200	0
LFR5	2000	31461	39	31.46	0.30	200	2
LFR6	2000	15495	32	15.49	0.15	200	3
LFR7	5000	70518	57	28.21	0.10	500	0
LFR8	5000	84899	61	33.96	0.25	500	2
LFR9	5000	93743	56	37.50	0.30	500	3
LFR10	10000	113290	61	22.66	0.10	1000	0
LFR11	10000	120047	75	24.01	0.20	1000	2
LFR12	10000	190440	84	38.09	0.25	1000	3

3.2. Parameter settings

The proposed algorithm is an adaptive algorithm by nature, meaning that the parameters can be tuned to achieve good community detection results. The configured parameters are the number of fast iterations in the first phase, the number of

refinement iterations, top- k labels, max labels, overlapping threshold, gain threshold, and gain factor. The fast iteration converged well within 8 iterations, while the modularity can be stabilised in 6 iterations. The top-3 neighbour labels is a good balance between noise and diversity, and limiting 3 labels per node helps avoid fragmentation. The low number of τ , ε , and η is to control the overlapping sets, prevent overfitting, and ensure effective propagation speed. The best average setting for the dataset used in this experiment is presented in Table 2.

Table 2. Parameter settings.

Name	Symbol	Value	Function/Role
Fast iterations	T_f	8	Number of label-voting passes
Refinement iterations	T_r	6	Number of modularity-refinement
Top- k labels	k	3	Number of neighbor can be considered
Max labels	L_{max}	3	Number of labels a node can hold
Overlapping threshold	τ	0.25	Weight threshold to allow a label into overlapping set
Gain threshold	ε	1e-4	Minimum modularity gain to allow a node to change label
Gain factor	η	0.6	Scaling factor for the neighboring votes
Random seed	-	75	Ensure the reproducibility

3.3. Modularity evaluation

The modularity measurement is a common and widely used method to evaluate community detection algorithms to show the density or sparsity of the results. The larger value indicates a densely connected structure within the results, but it does not indicate the accuracy of the community detection. The results presented in Table 3 show that compared to other algorithms, BD-LPA performed well and almost achieved the highest score for most datasets, with a peak score of 0.853 in LFR12. Motif-LPA performs competitively in synthetic datasets but struggles in real datasets, while SLPA and COPRA show moderate results. The average modularity across all datasets shows the superiority of BD-LPA with an average score of 0.647. This indicates BD-LPA is not only consistent but also adaptable to different network structures.

3.4. ONMI evaluation

The ONMI evaluation assesses the results of overlapping community detection algorithm against the ground truth of the datasets. The ONMI results of BD-LPA achieve the highest score on real-world datasets such as Karate, Football, and Polbooks. The results presented in Table 4 show that BD-LPA performs well for most

Table 3. Modularity score obtained from the datasets.

Dataset	SLPA	COPRA	Motif-LPA	BD-LPA
Karate	0.419	0.420	0.226	0.419
Football	0.520	0.530	0.557	0.571
Polbooks	0.370	0.380	0.415	0.457
LFR1.gml	0.803	0.802	0.802	0.897
LFR2.gml	0.000	0.000	0.494	0.580
LFR3.gml	0.000	0.000	0.713	0.718
LFR4.gml	0.501	0.000	0.509	0.519
LFR5.gml	0.665	0.667	0.668	0.662
LFR6.gml	0.000	0.000	0.736	0.749
LFR7.gml	0.529	0.000	0.532	0.536
LFR8.gml	0.610	0.595	0.610	0.624
LFR9.gml	0.827	0.811	0.829	0.812
LFR10.gml	0.000	0.000	0.613	0.621
LFR11.gml	0.000	0.000	0.689	0.689
LFR12.gml	0.000	0.000	0.840	0.853
Average	0.349	0.280	0.616	0.647

datasets, competing with Motif-LPA which performs strongly on synthetic datasets but shows poorly in real-world datasets. Considering its average performance, BD-LPA is superior to other algorithms with an average of 0.922 ONMI scores. This indicates that BD-LPA not only finds community with high modularity (as seen in Table 3) but also closely aligns its detected structure with ground truth across datasets.

3.5. Omega index

The omega index is specifically designed to handle overlapping community detection as the extension of the adjusted rand index (ARI) [2, 20]. This metric measures the similarity between the node memberships in predicted communities and the ground-truth communities. The range is from -1 to 1 , where 1 indicates perfect similarity, 0 means random communities, and -1 means that pairwise co-membership assignments are as discordant as possible relative to chance. Table 5 presents the omega index score from the datasets. BD-LPA performed well for the real-world datasets and almost all the synthetic datasets. BD-LPA consistently achieves high scores across datasets, notably reaching the highest scores for LFR6, LFR7, LFR8, LFR9, and LFR11. Motif-LPA performs competitively and reaches high scores in the synthetic dataset; however, it performed poorly on real-world datasets. Averaging across datasets, BD-LPA attains the highest mean omega index of 0.930 , indicating its ability in overlap-sensitive agreement metrics and showing its reliability across scenarios.

Table 4. ONMI score.

Dataset	SLPA	COPRA	Motif-LPA	BD-LPA
Karate	0.334	0.048	0.178	0.837
Football	0.283	0.283	0.339	0.771
Polbooks	0.296	0.429	0.444	0.569
LFR1.gml	0.892	0.946	1.000	0.947
LFR2.gml	0.649	0.838	0.980	0.986
LFR3.gml	0.725	0.624	0.990	0.981
LFR4.gml	0.743	0.876	0.998	0.929
LFR5.gml	0.799	0.860	1.000	0.939
LFR6.gml	0.750	0.672	0.980	0.999
LFR7.gml	0.780	0.865	0.998	0.987
LFR8.gml	0.805	0.859	0.999	0.983
LFR9.gml	0.857	0.826	1.000	0.950
LFR10.gml	0.806	0.892	1.000	0.981
LFR11.gml	0.777	0.681	0.990	1.000
LFR12.gml	0.814	0.748	1.000	0.965
Average	0.687	0.696	0.860	0.922

Table 5. Omega index score.

Dataset	SLPA	COPRA	Motif-LPA	BD-LPA
Karate	0.210	0.048	0.247	0.882
Football	0.671	0.080	0.536	0.778
Polbooks	0.546	0.700	0.568	0.637
LFR1.gml	0.954	0.425	1.000	0.977
LFR2.gml	0.612	-0.033	0.989	0.973
LFR3.gml	0.790	0.155	0.997	0.991
LFR4.gml	0.760	-0.024	0.998	0.936
LFR5.gml	0.905	0.244	1.000	0.964
LFR6.gml	0.855	0.166	0.985	1.000
LFR7.gml	0.905	-0.022	0.998	1.000
LFR8.gml	0.940	-0.003	1.000	1.000
LFR9.gml	0.972	0.287	1.000	1.000
LFR10.gml	0.957	0.002	1.000	0.900
LFR11.gml	0.946	0.184	0.991	1.000
LFR12.gml	0.972	0.281	1.000	0.957
Average	0.800	0.166	0.887	0.930

3.6. Generalized F1-score

The generalized F1-score compares the similarity between the predicted community with the ground-truth community. This metric focuses on quality rather than the structure of the compared communities. Table 6 presents the results of generalized F1-score across the datasets and shows that BD-LPA consistently deliver high scores across the datasets and reaches perfect alignment in LFR6. Its performance in real-world datasets is mixed, with an exceptional score of 0.971 in the Karate dataset but lower in other datasets; however, it performed exceptionally well in structured synthetic datasets. Motif-LPA also performed well in most datasets and reached perfect alignment in several cases. In terms of averages, BD-LPA leads with a mean score of 0.899, followed by Motif-LPA (0.837), COPRA (0.803), and SLPA (0.599). This indicates BD-LPA as the most accurate and balanced method across datasets, maintaining its strong precision-recall trade-offs.

Table 6. Generalized F1-score.

Dataset	SLPA	Copra	Motif-LPA	BD-LPA
Karate	0.609	0.600	0.278	0.971
Football	0.616	0.215	0.277	0.489
Polbooks	0.456	0.726	0.047	0.596
LFR1.gml	0.688	0.976	1.000	0.772
LFR2.gml	0.541	0.924	0.993	0.923
LFR3.gml	0.570	0.765	0.983	0.984
LFR4.gml	0.554	0.934	0.999	0.942
LFR5.gml	0.572	0.935	1.000	0.982
LFR6.gml	0.543	0.809	0.985	1.000
LFR7.gml	0.536	0.948	0.999	0.985
LFR8.gml	0.548	0.952	0.999	0.985
LFR9.gml	0.563	0.906	1.000	0.983
LFR10.gml	0.531	0.951	1.000	0.900
LFR11.gml	0.524	0.766	0.994	0.994
LFR12.gml	0.532	0.636	1.000	0.985
Average	0.559	0.803	0.837	0.899

3.7. Extended variation of information

This metric quantifies the amount of information lost and gained when the nodes move from one community to another. The extended VI score ranges from 0 upwards, where 0 means identical communities, while the larger value indicates the dissimilarity between predicted communities and the ground-truth. Table 7 presents the extended VI across the benchmark datasets. BD-LPA has the best extended VI score, indicating there is no discrepancy between its detected com-

munities and ground truth. The average score shows that BD-LPA is far superior to other algorithms, indicating its precision and consistency in producing ground-truth communities across diverse datasets.

Table 7. Extended VI score.

Dataset	SLPA	COPRA	Motif-LPA	BD-LPA
Karate	1.000	1.252	1.000	0.306
Football	1.000	3.276	1.000	0
Polbooks	1.000	0.954	1.000	0.364
LFR1.gml	0.008	0.012	0.000	0.000
LFR2.gml	0.0413	0.0431	0.004	0.000
LFR3.gml	0.028	0.125	0.003	0.000
LFR4.gml	0.018	0.029	0.000	0.000
LFR5.gml	0.012	0.037	0.000	0.000
LFR6.gml	0.016	0.111	0.005	0.000
LFR7.gml	0.007	0.024	0.000	0.000
LFR8.gml	0.005	0.024	0.000	0.000
LFR9.gml	0.004	0.033	0.000	0.000
LFR10.gml	0.003	0.014	0.000	0.000
LFR11.gml	0.004	0.048	0.001	0.000
LFR12.gml	0.003	0.026	0.000	0.000
Average	0.210	0.401	0.200	0.045

4. Conclusion and future research

In this paper, we propose an overlapping community detection method based on a label propagation algorithm (LPA) and betweenness centrality. In the presence of numerous communities, nodes that exhibit high betweenness are prospective overlapping nodes, which may lead to a reduced membership score within a community. Multilabel assignment are quickly applied to nodes using label propagation, and using the inverse betweenness as the weight for community assignment. The modularity refinement phase uses the detected communities as the base community to be evaluated with the modularity measurement. If the modularity of the node exceeds the threshold, the designation will be altered.

The utilisation of betweenness centrality as the voting mechanism provides a simple computation yet better structure for the proposed communities. The use of betweenness suppresses label flow across communities, resulting in higher modularity. This avoids false overlapping memberships since nodes with low betweenness are most likely to become a single community member node, thereby affecting the label purity, and improving ONMI scores. The limitation of a high betweenness score also contributes to pairwise consistency in co-membership across all pairs of

nodes. Betweenness also minimises label diffusion, which leads to reducing the label entropy for better performance.

Across evaluation metrics such as Modularity, ONMI, Omega Index, Generalized F1-score, and Extended VI, BD-LPA consistently performs well. It produces highly accurate and robust community structures compared to the ground truth, optimally balancing precision and recall. This enables effective and accurate identification of overlapping communities without sacrificing detection sensitivity while minimising information loss between detected and actual community structures.

This study used parameter settings that were adjusted for all datasets and produced good results across all evaluations. For specific implementation tasks, such as biological networks, social networks, or communication systems, additional parameter settings may need to be adjusted to optimise community detection tasks achieve better results. Overall, these results suggest that this approach is particularly powerful in enhancing community detection quality for structured networks, although further refinements could improve accuracy on datasets with ambiguous or noisy metadata.

Future work on this algorithm could focus on three key areas. First, **real-world network adaptation** should be conducted, as performance on irregular graphs suggests potential sensitivity to noisy or incomplete structures. Second, **the scalability and efficiency** need to be explored for handling massive networks with millions of nodes potentially through parallelised or distributed implementation. Third, **dynamic and temporal network extension** offers a promising direction for the algorithm to track evolving overlapping communities over time, which is potentially useful for applications in social media, biological networks, and communication systems. Additionally, integration with a graph neural network pipeline could allow the algorithm to serve as a high-quality label generator or preprocessor for deep learning-based community detection.

References

- [1] J.-P. ATTAL, M. MALEK, M. ZOLGHADRI: *Overlapping community detection using core label propagation algorithm and belonging functions*, Applied Intelligence 51.11 (2021), pp. 8067–8087, DOI: [10.1007/s10489-021-02250-4](https://doi.org/10.1007/s10489-021-02250-4).
- [2] L. M. COLLINS, C. W. DENT: *Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions*, Multivariate behavioral research 23.2 (1988), pp. 231–242, DOI: [10.1207/s15327906mbr2302_6](https://doi.org/10.1207/s15327906mbr2302_6).
- [3] W. FANG, X. WANG, L. LIU, Z. WU, S. TANG, Z. ZHENG: *Community detection through vector-label propagation algorithms*, Chaos, Solitons & Fractals 158 (2022), p. 112066, DOI: [10.1016/j.chaos.2022.112066](https://doi.org/10.1016/j.chaos.2022.112066).
- [4] S. FORTUNATO, D. HRIC: *Community detection in networks: A user guide*, Physics reports 659 (2016), pp. 1–44, DOI: [10.1016/j.physrep.2016.09.002](https://doi.org/10.1016/j.physrep.2016.09.002).
- [5] L. C. FREEMAN: *A set of measures of centrality based on betweenness*, Sociometry (1977), pp. 35–41, DOI: [10.2307/3033543](https://doi.org/10.2307/3033543).
- [6] L. C. FREEMAN: *Centrality in social networks conceptual clarification*, Social networks 1.3 (1978), pp. 215–239.

- [7] S. GOSWAMI, A. K. SINGH: *A survey on overlapping community detection: label propagation*, Multimedia Tools and Applications (2024), pp. 1–30, DOI: [10.1007/s11042-024-20485-4](https://doi.org/10.1007/s11042-024-20485-4).
- [8] S. GREGORY: *Fuzzy overlapping communities in networks*, Journal of Statistical Mechanics: Theory and Experiment 2011.02 (2011), P02017, DOI: [10.1088/1742-5468/2011/02/P02017](https://doi.org/10.1088/1742-5468/2011/02/P02017).
- [9] P. W. HOLLAND, K. B. LASKEY, S. LEINHARDT: *Stochastic blockmodels: First steps*, Social networks 5.2 (1983), pp. 109–137, DOI: [10.1016/0378-8733\(83\)90021-7](https://doi.org/10.1016/0378-8733(83)90021-7).
- [10] V. KREBS: *Books about US politics*, unpublished, <http://www.orgnet.com> (2004).
- [11] A. KUMARI, A. KUMAR, P. KUMAR: *A new modularity metric for disjoint and overlapping community structure evaluation in weighted complex networks*, International Journal of Data Science and Analytics (2025), pp. 1–25.
- [12] A. LANCICHINETTI, S. FORTUNATO, F. RADICCHI: *Benchmark graphs for testing community detection algorithms*, Physical Review E—Statistical, Nonlinear, and Soft Matter Physics 78.4 (2008), p. 046110, DOI: [10.1103/PhysRevE.78.046110](https://doi.org/10.1103/PhysRevE.78.046110).
- [13] P.-Z. LI, L. HUANG, C.-D. WANG, J.-H. LAI, D. HUANG: *Community detection by motif-aware label propagation*, ACM Transactions on Knowledge Discovery from Data (TKDD) 14.2 (2020), pp. 1–19, DOI: [10.1145/3378537](https://doi.org/10.1145/3378537).
- [14] X. LI, Q. SUN: *Detect Overlapping Community Based on the Combination of Local Expansion and Label Propagation*, Algorithms 14.8 (2021), p. 237, DOI: [10.3390/a14080237](https://doi.org/10.3390/a14080237).
- [15] Y. LI, K. HE, D. BINDEL, J. E. HOPCROFT: *Uncovering the small community structure in large networks: A local spectral approach*, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 658–668, DOI: [10.1145/2736277.2741676](https://doi.org/10.1145/2736277.2741676).
- [16] Z. LIN, X. ZHENG, N. XIN, D. CHEN: *CK-LPA: Efficient community detection algorithm based on label propagation with community kernel*, Physica A: Statistical Mechanics and its Applications 416 (2014), pp. 386–399, DOI: [10.1016/j.physa.2014.09.023](https://doi.org/10.1016/j.physa.2014.09.023).
- [17] X. LIU, T. MURATA: *Advanced modularity-specialized label propagation algorithm for detecting communities in networks*, Physica A: Statistical Mechanics and its Applications 389.7 (2010), pp. 1493–1500, DOI: [10.1016/j.physa.2009.12.019](https://doi.org/10.1016/j.physa.2009.12.019).
- [18] A. F. MCDAID, D. GREENE, N. HURLEY: *Normalized mutual information to evaluate overlapping community finding algorithms*, arXiv preprint arXiv:1110.2515 (2011), DOI: [10.48550/arXiv.1110.2515](https://doi.org/10.48550/arXiv.1110.2515).
- [19] M. MEILÄ: *Comparing clusterings by the variation of information*, in: Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24–27, 2003. Proceedings, Springer, 2003, pp. 173–187, DOI: [10.1007/978-3-540-45167-9_14](https://doi.org/10.1007/978-3-540-45167-9_14).
- [20] G. MURRAY, G. CARENINI, R. NG: *Using the omega index for evaluating abstractive community detection*, in: Proceedings of workshop on evaluation metrics and system comparison for automatic summarization, 2012, pp. 10–18.
- [21] M. E. NEWMAN, M. GIRVAN: *Finding and evaluating community structure in networks*, Physical review E 69.2 (2004), p. 026113, DOI: [10.1103/PhysRevE.69.026113](https://doi.org/10.1103/PhysRevE.69.026113).
- [22] Q. NI, J. WANG, Z. TANG: *Degree and betweenness-based label propagation for community detection*, Journal of Combinatorial Optimization 49.2 (2025), p. 21, DOI: [10.1007/s10878-024-01254-3](https://doi.org/10.1007/s10878-024-01254-3).
- [23] G. PALLA, I. DERÉNYI, I. FARKAS, T. VICSEK: *Uncovering the overlapping community structure of complex networks in nature and society*, nature 435.7043 (2005), pp. 814–818, DOI: [10.1038/nature03607](https://doi.org/10.1038/nature03607).
- [24] C. PIZZUTI: *Ga-net: A genetic algorithm for community detection in social networks*, in: International conference on parallel problem solving from nature, Springer, 2008, pp. 1081–1090, DOI: [10.1007/978-3-540-87700-4_107](https://doi.org/10.1007/978-3-540-87700-4_107).

- [25] U. N. RAGHAVAN, R. ALBERT, S. KUMARA: *Near linear time algorithm to detect community structures in large-scale networks*, Physical Review E—Statistical, Nonlinear, and Soft Matter Physics 76.3 (2007), p. 036106, DOI: [10.1103/PhysRevE.76.036106](https://doi.org/10.1103/PhysRevE.76.036106).
- [26] A. SARSWAT, V. JAMI, R. M. R. GUDDATI: *A novel two-step approach for overlapping community detection in social networks*, Social Network Analysis and Mining 7.1 (2017), p. 47, DOI: [10.1007/s13278-017-0469-7](https://doi.org/10.1007/s13278-017-0469-7).
- [27] H.-W. SHEN, X.-Q. CHENG, J.-F. GUO: *Quantifying and identifying the overlapping community structure in networks*, Journal of Statistical Mechanics: Theory and Experiment 2009.07 (2009), P07042, DOI: [10.1088/1742-5468/2009/07/P07042](https://doi.org/10.1088/1742-5468/2009/07/P07042).
- [28] F. WANG, T. LI, X. WANG, S. ZHU, C. DING: *Community discovery using nonnegative matrix factorization*, Data Mining and Knowledge Discovery 22.3 (2011), pp. 493–521, DOI: [10.1007/s10618-010-0181-y](https://doi.org/10.1007/s10618-010-0181-y).
- [29] J. XIE, B. K. SZYMANSKI, X. LIU: *Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process*, in: 2011 IEEE 11th International Conference on Data Mining Workshops, IEEE, 2011, pp. 344–349, DOI: [10.1109/ICDMW.2011.154](https://doi.org/10.1109/ICDMW.2011.154).
- [30] H. XU, Y. RAN, J. XING, L. TAO: *An influence-based label propagation algorithm for overlapping community detection*, Mathematics 11.9 (2023), p. 2133, DOI: [10.3390/math11092133](https://doi.org/10.3390/math11092133).
- [31] M. YAN, C. GUOQIANG: *Label propagation community detection algorithm based on density peak optimization*, in: 2021 17th International Conference on Computational Intelligence and Security (CIS), IEEE, 2021, pp. 80–84, DOI: [10.1109/CIS54983.2021.00025](https://doi.org/10.1109/CIS54983.2021.00025).
- [32] J. YANG, J. LESKOVEC: *Overlapping community detection at scale: a nonnegative matrix factorization approach*, in: Proceedings of the sixth ACM international conference on Web search and data mining, 2013, pp. 587–596, DOI: [10.1145/2433396.2433471](https://doi.org/10.1145/2433396.2433471).
- [33] S. YAZDANPARAST, M. JAMALABDOLLAHI, T. C. HAVENS: *Linear time community detection by a novel modularity gain acceleration in label propagation*, IEEE Transactions on Big Data 7.6 (2020), pp. 961–966, DOI: [10.1109/TBDATA.2020.2995621](https://doi.org/10.1109/TBDATA.2020.2995621).
- [34] W. ZHANG, R. SHANG, L. JIAO: *Large-scale community detection based on core node and layer-by-layer label propagation*, Information Sciences 632 (2023), pp. 1–18, DOI: [10.1016/j.ins.2023.02.090](https://doi.org/10.1016/j.ins.2023.02.090).

Syntactic comparison of human and AI-written scientific texts

Erika B. Varga^a, Attila Baksa^b

^aInstitute of Information Technology
erika.b.varga@uni-miskolc.hu

^bInstitute of Applied Mechanics
Faculty of Mechanical Engineering and Informatics
University of Miskolc, H-3515 Miskolc, Hungary
attila.baksa@uni-miskolc.hu

Abstract. The spread of large language models (LLMs) has transformed scientific writing, enabling the generation of fluent and convincing text with minimal human input. This development poses significant challenges for authorship verification, especially when AI-generated or AI-assisted content is embedded in academic manuscripts. While most existing detection approaches rely on surface-level lexical features or stylometric clues, our study proposes a novel syntactic-level method to distinguish between human-authored, translated, and AI-generated scientific texts. We constructed a controlled corpus of 24 scientific articles in the field of computer science, divided into four categories: native-authored, human-translated, ChatGPT 4.0-generated, and ChatGPT 4o-generated with deep research. Each corpus was processed using part-of-speech (POS) and dependency parsing, followed by statistical profiling and sentence-structure discovery via process mining. Our results reveal that AI-generated texts differ significantly in their use of modal verbs, participles, coordination, and syntactic complexity. We demonstrate that process-mined graphs of syntactic transitions provide an interpretable and robust fingerprint of authorship, enabling us to detect AI-generated patterns and differentiate them from translated or native writing. The proposed framework contributes a novel methodological perspective to the growing field of AI authorship detection.

Keywords: AI-generated text detection, syntactic analysis, sentence structure modeling, process discovery

AMS Subject Classification: 68T50, 68T20, 68U15

1. Introduction

AI tools built on large language models (LLMs) such as ChatGPT have become increasingly widespread in academic writing. From brainstorming and drafting to paraphrasing and summarization, LLMs are now embedded in the workflow of researchers and students alike. This shift in authorship practices has raised important questions about the authenticity of scientific writing and the ability to distinguish AI-generated text from human-authored content.

Current approaches to AI detection focus predominantly on surface-level characteristics such as word frequency patterns, perplexity scores, and stylometric irregularities [9, 16]. While these methods offer fast and scalable detection, AI-generated texts can slip through these systems by means of paraphrasing, translation, or editing, which obscure the statistical fingerprints of AI models. Moreover, stylometric features tend to treat grammatical elements as isolated tokens rather than analyzing how sentences are structured syntactically [6].

In this paper, we argue that syntactic patterns, especially sentence-level grammatical structures, offer a deeper and more robust basis for detecting AI-generated writing. We focus on sentence structure as a distinctive linguistic fingerprint and investigate how it varies across different types of text: native-authored, human-translated, ChatGPT-generated, and ChatGPT 4o-generated (in deep research mode). To this end, we created a four-part scientific text corpus, each group consisting of six scientific papers written or generated under controlled conditions in the computer science domain. We apply two complementary methods: (1) statistical profiling based on part-of-speech (POS) tags and dependency roles, and (2) process mining to discover generalized syntactic flow patterns in sentence construction using the Heuristics Miner algorithm [14]. While the former quantifies grammatical characteristics, the latter visualizes structural tendencies through heuristic process graphs. This experiment introduces a novel application of process mining in linguistic analysis.

2. Related works

Detecting AI-generated or AI-influenced text has become a prominent research field, particularly with the widespread use of large language models (LLMs) in academic writing. Stylometric methods relying on features such as function-word usage, part-of-speech (POS) distributions, and sentence-length metrics are proving effective in distinguishing AI-generated content from human-authored text. Notably, Zaitzu & Jin [16] achieved 98 % accuracy in classifying GPT-generated versus human-written Japanese scientific text using POS bigrams and function-word frequencies. Prova (2024) [9] presents a hybrid detection model using feature-based classifiers (XGBoost, SVM) alongside a BERT-based architecture. Trained on a balanced dataset of AI- and human-generated samples, the BERT model achieved 93 % accuracy, outperforming XGBoost (84 %) and SVM (81 %). The recent StyloAI model [6] applies 31 stylometric features, including new grammatical markers,

using Random Forest classifier and achieves up to 98 % accuracy in multi-domain detection tasks.

However, these approaches treat syntactic structure as static features, such as isolated POS or dependency tag frequencies. Researchers have advanced the field by using POS n-grams and syntactic n-grams. For instance, Pokou et al. in [7] introduced variable-length POS patterns for authorship attribution, capturing frequent POS sequences rather than single-tag statistics. Posadas-Durán et al. in [8] used complete syntactic n-grams from dependency trees to profile writing style, demonstrating improved performance over lexical n-grams. Still, these methods remain static, focusing on patterns without modeling the sequential structure of syntactic roles.

Structured dependency stylometry has also gained attention. Murauer & Specht in [5] proposed DT-grams, language-independent dependency-tree-gram features for cross-language authorship identification, proving their utility in capturing grammatical style when transferring texts across languages. However, like POS and syntactic n-grams, DT-grams treat dependency substructures in isolation and do not model the flow of syntactic roles within sentences.

In parallel, AI-influenced translation, where human text is revised or translated by LLMs, poses new detection challenges. Systems like GPTZero struggle to identify AI-assisted rewriting, with performance dropping to 28 % F1 in paraphrased cases and producing false positives in human writing [3]. Krishna et al. in [4] report similar challenges, noting that AI-assisted paraphrasing often avoids detection due to preservation of human syntactic patterns.

Our earlier work addresses part of this challenge through lexical profiling. In [13], we introduced a synonym set based approach using WordNet synsets to measure conceptual recursion and redundancy. We found that translated texts exhibit higher lexical density than AI texts, while native texts share greater concept overlap with AI output. We also analyzed lexical redundancy and conceptual overlap, showing clear distinctions between AI- and human-authored scientific writing in [12].

While stylometry outlines linguistic fingerprints, it does not capture the dynamic flow of sentence construction. Process mining has been recently combined with NLP for extracting structured models from event logs, with applications such as semantic role labeling from textual logs [10] and exploratory process model discovery via language models [11]. However, no prior work directly applies process mining to syntactic dependency sequences in text.

Our present approach applies Heuristic Miner to build dependency-role transition graphs from sentence-level parse sequences, yielding graphical models that reflect typical structures. This method extends traditional syntactic analysis by representing how roles (e.g., subject $\rightarrow$ verb $\rightarrow$ object $\rightarrow$ modifier) sequentially co-occur, offering a novel, process-oriented view. To our knowledge, it is the first attempt at such structural profiling. In the context of translated texts, which may combine human sentence planning with AI-driven surface-level phrasing, our process-mining approach reveals patterns such as syntactic chaining or padding

that are indicative of AI involvement. These structural patterns supplement lexical measures, enabling a more robust detection of AI assistance in translation.

3. Methodology

3.1. Corpus design

The text corpus used in this study consists of 24 scientific articles from the Computer Science domain, divided into four categories, each containing six texts. The first category includes human-written papers authored by native English speakers and published between 2000 and 2015, prior to the emergence of LLMs. Their counterparts form the second category: articles generated by ChatGPT-4. For these, the title and abstract of each human-written paper were provided as prompts, together with an instruction to write a full scientific article of about 10,000 words. In practice, however, ChatGPT-4 produced outputs of approximately 5,000 words, and none of the generated texts exceeded this length.

The third category consists of translations of papers written by non-native researchers after 2020. These texts were created with moderate AI support, which was used only to improve linguistic quality, such as correcting grammatical errors, smoothing the discussion flow, and enhancing stylistic variety. The fourth category contains the AI-generated counterparts of these translations, produced with ChatGPT-4o's deep research function. As in the previous case, the model was given the title and abstract of each translated paper and instructed to write a full scientific article of about 10,000 words. In this case, ChatGPT-4o complied with the requested length and generated significantly longer articles, typically over 10,000 words.

The use of these two different AI models explains the variation in text length between the paired categories. All texts nevertheless exceed 3,000 words, ensuring sufficient syntactic complexity and topic depth for the analysis. The basic characteristics of the papers in the collection, as well as their lexical comparison, are reported in a recent paper [13].

3.2. Syntactic annotation

Text files were first cleaned and segmented into individual sentences using a combination of regular expressions and the NLTK python library's sentence tokenization function. Non-linguistic elements such as titles, metadata, and references were excluded. Sentences were then normalized by: (1) removing punctuation, numeric tokens, and special characters; (2) filtering short, non-informative lines (e.g., headings) and retaining sentences with sufficient lexical and grammatical complexity. Next, each sentence was POS-tagged and dependency-parsed using spaCy's `en_core_web_sm` model to enable sentence-level syntactic profiling.

3.3. Sentence structure discovery

In this study, we introduce a novel methodological contribution by applying process mining techniques to explore and visualize the structural complexity of natural language sentences. Traditionally used in business process management and system logs, process discovery allows the extraction of structured workflow models from event logs [2]. Here, we adapt this technique to syntactic dependency data, treating each sentence as a trace and each dependency label as an event within that trace. For this approach, all texts had to be transformed into an event log that captures the sequence of syntactic roles. To illustrate the concept, a simplified example is shown below:

$$\mathcal{L} = \{\langle A, B, C, D \rangle, \langle A, C, D \rangle, \langle A, B, D \rangle\}$$

In this example, log $\mathcal{L}$ contains three traces, each representing a sentence as an ordered list of dependency labels. A , B , C , and D are placeholder symbols standing for different dependency labels that represent syntactic roles (e.g., subject, verb, object, modifier).

Event logs typically contain at least a case ID (a unique identifier for a process instance), an activity label (representing an event), and a timestamp. In our setting, the timestamps do not carry linguistic information but are required by the process mining algorithm to determine the order of events within each trace. They were therefore generated automatically according to the sequential order of the dependency labels in a sentence. The syntactic annotation and log generation pipeline is shown in Figure 1.

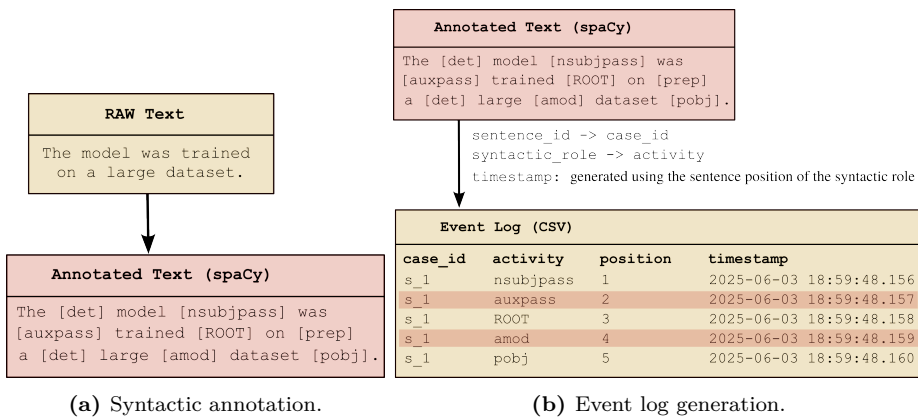


Figure 1. Text transformation into event log.

During the construction of the event logs, not all syntactic roles (dependency labels) were considered equally relevant. To focus the analysis on core sentence structure and reduce noise from frequent but less informative elements, several dependency types were deliberately excluded from the logs. The omitted labels include common grammatical markers such as determiners (det), prepositions and

case markers (case), subordinating and coordinating conjunctions (mark, cc), pre-conjunctions (preconj) and punctuation (punct). These roles typically represent grammatical scaffolding rather than semantic or structural functions within a sentence. By excluding them, the resulting log better captures the backbone of syntactic constructions, such as subject–verb–object relationships and clause-level dependencies.

Common process discovery algorithms include (1) Alpha Miner [1], that is suitable for simple, noise-free logs, (2) Heuristics Miner [14] which is robust against exceptional or infrequent behavior, and (3) Inductive Miner [15] which produces sound, block-structured models. For this study, the Heuristics Miner algorithm was selected for two main reasons. Firstly, natural language sentence structures exhibit high variability. Nearly every sentence can be seen as a unique case with different syntactic sequences. This results in a highly exceptional log with few frequent patterns. The Heuristics Miner is designed to handle such noisy, non-repetitive event logs, making it ideal for linguistic applications. Secondly, unlike other algorithms that produce complex Petri nets, the Heuristics Miner generates a graphical model, called heuristic net, that emphasizes the most statistically significant transitions between events. This representation is more interpretable for syntactic analysis, where the goal is not strict process conformance but insight into sentence construction tendencies like how subjects relate to objects, or the placement of modifiers.

4. Results

This section presents the results of sentence-level grammatical profiling across native-authored, translated, and AI-generated scientific texts. The analysis addresses two main objectives:

1. to quantify the statistical differences in grammatical structure between the text categories, and
2. to identify distinctive sentence structure patterns that can differentiate human-authored and translated texts from AI-generated ones.

4.1. Quantitative analysis of grammatical differences

To quantify the differences between the four groups of texts, we have applied four statistical measures. All grammatical comparisons are based on normalized or sentence-level average metrics to eliminate distortions caused by differing document lengths.

4.1.1. POS distribution

The normalized POS tag frequencies in Figure 2 reveal clear lexical patterns that differentiate the three text groups. Notably, AI-generated texts stand out for their higher frequency of modal verbs (MD), present participles (VBG), adjectives (JJ),

plural nouns (NNS), and adverbs (RB). These patterns suggest a tendency toward generalization, elaboration, and structural padding. These are typical features of model-generated language that aims for academic style without strong referential grounding. Native-authored texts lead in the use of proper nouns (NNP), infinitival markers (TO), and personal pronouns (PRP), reflecting a more referential and agent-oriented writing style. Not surprisingly, translated texts share features with both groups. They resemble AI texts in their frequent use of common nouns (NN, NNS) and TO, which suggests compact syntax likely arising from translation conventions or simplification. On the other hand, they align more closely with native texts in their use of past participles (VBN) and proper nouns (NNP), the features associated with academic conventions like passive constructions and citations.

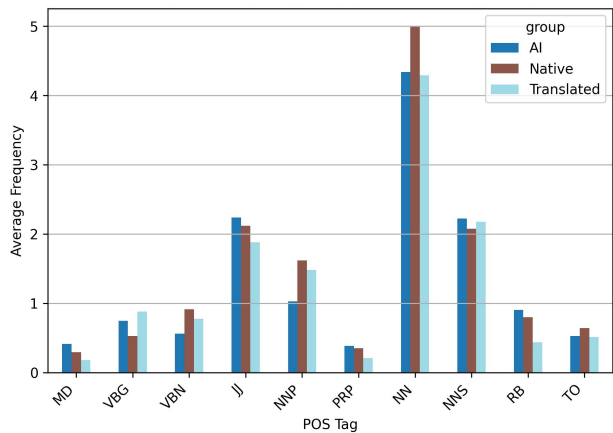


Figure 2. Average POS distributions.

4.1.2. Syntactic role distribution

The syntactic structure of sentences was analyzed by grouping dependency roles into higher-level categories (e.g., *Subject*, *Object*, *Modifier*). The radar chart in Figure 3 visualizes the average number of each role per sentence, aggregated by text group. Remarkably, AI-generated and native-authored texts show similar syntactic role distributions across most categories. This suggests that AI writing tools successfully replicate human-like sentence structuring in scientific text. The most notable divergence is seen in the Modifier role. Human-written texts consistently use more modifiers, indicating a tendency to enrich noun phrases or insert descriptive elements. AI texts, by contrast, apply modifiers more conservatively, potentially favoring structural clarity over elaboration. In terms of Coordination, the pattern is reversed. AI texts exhibit higher coordination than human-written ones, suggesting a preference for parallel or additive structures. Native texts rely less on coordination, possibly favoring more hierarchical or subordinated constructions. The Object role also shows a distinct contrast. Native-authored texts feature

the highest object frequency, which may reflect more diverse transitive constructions or greater syntactic density. AI texts exhibit lower values here, hinting at more simplified or formulaic sentence building. Other roles such as *Subject*, *Predicate*, and *Adverbial* remain relatively stable across all groups, indicating a shared core structure in scientific writing regardless of authorship.

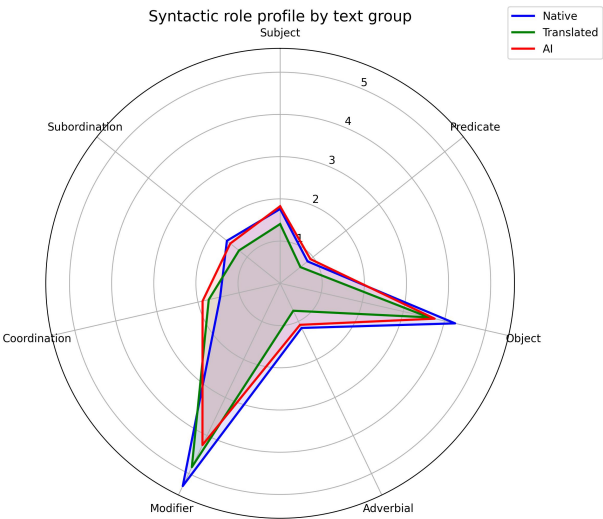


Figure 3. Average distribution of syntactic roles.

4.1.3. Position of key syntactic roles

The most frequent positions of the main syntactic roles (*nsubj*, *ROOT*, *dobj*) shown in Figure 4 further distinguish the groups. Although native texts have the longest sentences, averaging over 25 tokens, the subject (*nsubj*) typically appears in the first position, while the predicate (*ROOT*) appears in the second position, which aligns with canonical English word order (*Subject–Verb–Object*). In AI-generated texts, these roles are delayed, likely due to the frequent use of fronted modifiers. Direct object (*dobj*) shows the most notable difference. In AI-generated texts, it appears later (usually at position 7), while in both native and translated texts the object appears at position 5. This suggests that AI systems tend to insert more modifiers between the verb and the object, creating syntactically padded structures.

4.1.4. Sentence complexity profiles

The comparison of normalized sentence structure distributions in Figure 5 highlights distinct tendencies across author groups. Interestingly, AI-generated texts rely heavily on active constructions, with nearly 80 % of sentences being active.

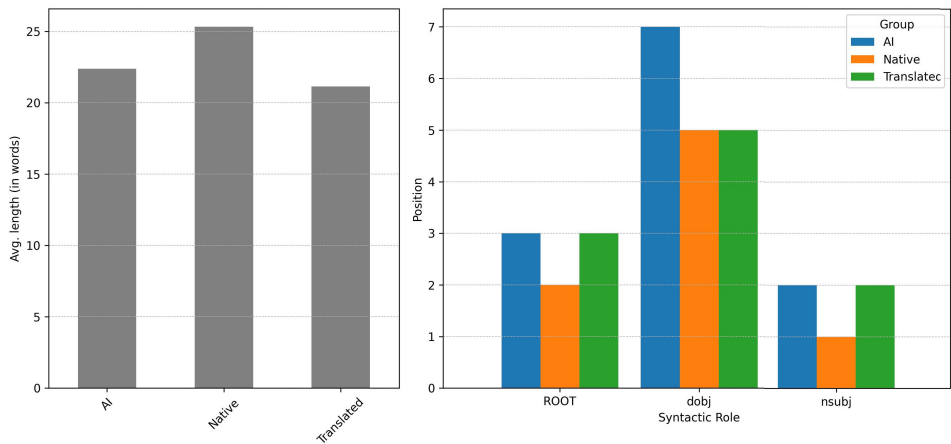


Figure 4. (a) Average sentence length (b) Most frequent positions of key syntactic roles.

They also show minimal use of complex and passive structures, suggesting syntactic simplicity and regularity. This reflects the model’s preference for clarity and reduced grammatical embedding. Native-authored texts, in contrast, display greater structural diversity. While active voice still dominates, these texts feature more passive and complex sentences, indicating a higher degree of syntactic flexibility and subordination. Translated texts exhibit a hybrid behavior. Their use of complex and fragmentary sentences is slightly higher than in native writing, possibly due to literal rendering of source syntax or segmentation artifacts introduced by translation tools. Passive constructions occur less frequently than in native texts, but more than in AI-generated content, pointing to some retention of authentic academic style.

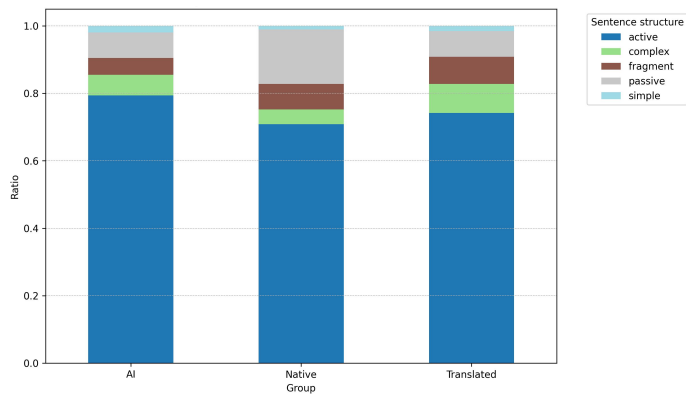


Figure 5. Normalized sentence structure ratios.

While the statistical profiles above capture frequency-based syntactic tendencies, process mining allows us to explore how these grammatical elements are sequenced and interact within individual sentences.

4.2. Sentence structure discovery

To identify structural patterns typical of each author group, we applied the Heuristic Miner algorithm to generate dependency-role process graphs for every single text in the corpus. These individual graphs model the transitions between major dependency roles, capturing both their frequency and structural positioning within sentences. We also experimented with creating aggregated process graphs for entire groups; however, these models became overly large and heterogeneous, making them less suitable for meaningful interpretation. For this reason, the paper presents one representative graph from each group, selected to illustrate characteristic sentence-level syntactic tendencies identified in that category. Specifically, Figure 6a shows an example from the human-translated texts, Figure 6b from the ChatGPT-4 generated texts, Figure 7a from the native-authored texts, and Figure 7b from the ChatGPT-4o deep research texts. The development of more effective methods for constructing generalized group-level process models is left for future work.

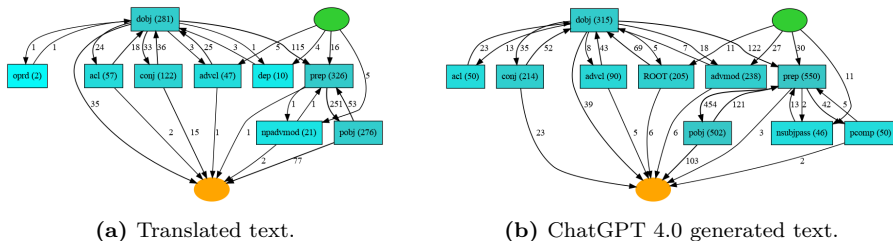


Figure 6. Heuristic dependency structure graphs of translated and ChatGPT 4.0-generated texts.

We can observe that the process graphs of human-translated and ChatGPT 4.0 texts are structurally similar. Both graphs display relatively shallow syntactic structures which indicates a tendency toward simpler and flatter clause chains. These texts exhibit frequent use of prepositional phrases which reflects compact sentence construction and dense nominal modification (i.e., sequences of stacked adjectives or noun modifiers within noun phrases). This is considered typical in translated texts and baseline AI. The lower presence of subordinate structures (`xcomp`, `advcl`, and `csbj`) suggests limited clause embedding. It is also worth noting, that both groups rely heavily on direct object constructions, indicating strong SVO alignment and topic-focus in sentence planning.

The process graphs of native writings and texts generated by ChatGPT 4o deep research function show deeper syntactic complexity and stronger structural variation. Subordinate clause structures (`advcl`, `xcomp`, and `acl`) appear more often

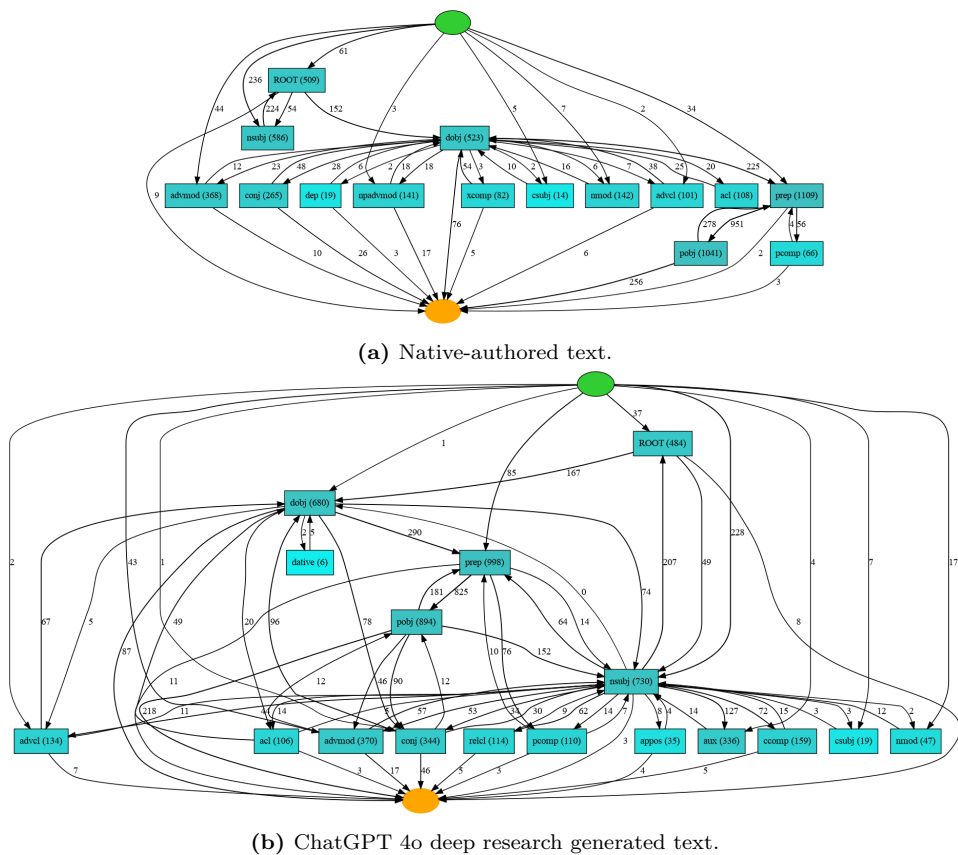


Figure 7. Heuristic dependency structure graphs of native and ChatGPT 4o deep research generated texts.

and in richer configurations. Especially, subject and object (**nsubj** and **dobj**) are embedded more frequently inside these clauses, which indicates syntactic elaboration and hierarchical depth. Also, both graphs include more dependency roles (**conj** and **ccomp**) and more transitions, which suggests the high use of coordinated or parallel clause structures.

5. Discussion and conclusion

This study set out to identify grammatical features that can help infer the extent of AI involvement in human-translated scientific texts. Through a combination of frequency-based syntactic profiling and structural process mining, we have uncovered clear patterns that distinguish AI-generated content from both native-authored and translated texts.

Our analysis revealed that AI-generated texts exhibit distinctive surface-level tendencies. These include an elevated use of modal verbs, participles, and coordinated structures, combined with reduced use of passive voice and embedded clauses. Although these patterns aim to mimic academic tone, they often result in overly regular sentence structures with less syntactic depth. By contrast, native-authored texts demonstrated more referential grounding and syntactic variability, marked by higher frequencies of proper nouns, passive constructions, and subordinate clauses.

Not surprisingly, translated texts emerged as a hybrid category. In many aspects, such as the prevalence of common nouns, infinitival markers, and prepositional phrases, they aligned with baseline AI output. This suggests the use of translation systems which tend to apply simplification and structural compression. However, their use of passive voice, proper nouns, and past participles more closely resembled human writing, particularly in stylistic conventions of scientific discourse.

Notably, the most insightful patterns emerged from process mining. The structural graphs showed that baseline AI (ChatGPT 4.0 default) and translated texts rely on flatter syntactic chains with limited embedding, while native and advanced AI (ChatGPT 4o with deep research) texts reveal richer clause interactions which reflect a deeper level of syntactic planning.

These findings suggest that it is possible to develop diagnostic criteria to detect AI involvement in translated texts. Texts that display (1) high coordination but low subordination, (2) frequent direct object placement at later sentence positions, (3) dense nominal modification with limited clause embedding, and (4) reduced use of referential markers (e.g., proper nouns, personal pronouns), are more likely to contain AI-generated segments.

While translated texts may inherit some of these attributes from the source language or the translation process itself, an over-concentration of such features, especially when aligned with statistical outliers in modifier usage or syntactic role position, can serve as a heuristic indicator of AI usage.

6. Limitations and future work

A limitation of the present study is that syntactic roles were automatically assigned using the `spaCy` parser, which may introduce annotation inaccuracies. Another limitation concerns the corpus size: with 24 papers divided into four categories, the dataset is adequate for a proof-of-concept but relatively small for drawing broader conclusions. While each paper is long enough to provide syntactic depth, the limited number of texts constrains the representativeness of the findings. As a direction for future research, we aim to expand the corpus and to develop methods for constructing more interpretable generalized process graphs at the group level, so that broader structural tendencies can be captured.

References

- [1] W. VAN DER AALST, T. WEIJTERS, L. MARUSTER: *Workflow mining: discovering process models from event logs*, IEEE Transactions on Knowledge and Data Engineering 16.9 (2004), pp. 1128–1142, DOI: [10.1109/TKDE.2004.47](https://doi.org/10.1109/TKDE.2004.47).
- [2] W. VAN DER AALST: *Process Mining*, Springer Berlin, Heidelberg, 2016, ISBN: 978-3-662-49850-7, DOI: [10.1007/978-3-662-49851-4](https://doi.org/10.1007/978-3-662-49851-4).
- [3] D. BROWN, D. JENSEN: *GPTZero vs. Text Tampering: The Battle that GPTZero Wins*, in: ed. by M. SHELLEY, V. AKERSON, M. UNAL, International Conference on Social and Education Sciences, Las Vegas, NV, USA. ISTES Organization, 2023, pp. 734–744, URL: <https://files.eric.ed.gov/fulltext/ED656089.pdf>.
- [4] K. KRISHNA, Y. SONG, M. KARPINSKA, J. WIETING, M. IYER: *Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense*, 2023, URL: <https://arxiv.org/abs/2303.13408>.
- [5] B. MURAUER, G. SPECHT: *DT-grams: Structured Dependency Grammar Stylometry for Cross-Language Authorship Attribution*, 2021, URL: <https://arxiv.org/abs/2106.05677>.
- [6] C. OPARA: *StyloAI: Distinguishing AI-Generated Content with Stylometric Analysis*, 2024, URL: <https://arxiv.org/abs/2405.10129>.
- [7] Y. J. M. POKOU, P. FOURNIER-VIGER, C. MOGHRABI: *Authorship Attribution using Variable Length Part-of-Speech Patterns*, in: Proceedings of the 8th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART, INSTICC, SciTePress, 2016, pp. 354–361, ISBN: 978-989-758-172-4, DOI: [10.5220/0005710103540361](https://doi.org/10.5220/0005710103540361).
- [8] J.-P. POSADAS-DURAN, G. SIDOROV, I. BATYRSHIN: *Complete Syntactic N-grams as Style Markers for Authorship Attribution*, in: Human-Inspired Computing and Its Applications, ed. by A. GELBUKH, F. C. ESPINOZA, S. N. GALICIA-HARO, Cham: Springer International Publishing, 2014, pp. 9–17, ISBN: 978-3-319-13647-9, DOI: [10.1007/978-3-319-13647-9_2](https://doi.org/10.1007/978-3-319-13647-9_2).
- [9] N. PROVA: *Detecting AI Generated Text Based on NLP and Machine Learning Approaches*, 2024, URL: <https://arxiv.org/abs/2404.10032>.
- [10] A. REBMANN, H. VAN DER AA: *Extracting Semantic Process Information from the Natural Language in Event Logs*, 2021, URL: <https://arxiv.org/abs/2103.11761>.
- [11] A. REBMANN, F. D. SCHMIDT, G. GLAVAŠ, H. VAN DER AA: *Evaluating the Ability of LLMs to Solve Semantics-Aware Process Mining Tasks*, 2024, URL: <https://arxiv.org/abs/2407.02310>.
- [12] E. B. VARGA, A. BAKSA: *A Comparative Analysis of Human and Machine Written Scientific Texts*, in: ed. by E. XHAFERRA, Proceedings of the 1st International Conference on Computer Science and Information Technology, Durrës, Albania, 2025, URL: https://csitconf.org/Book_of_Proceedings_CSIT2025.pdf.
- [13] E. VARGA, A. BAKSA: *Exporing Lexical Variation Through Synonym Sets in Human and AI-Written Scientific Texts*, submitted to journal Multidisciplinary Sciences, 2025.
- [14] A. WEIJTERS, W. AALST, A. MEDEIROS: *Process Mining with the Heuristics Miner-algorithm*, BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006.
- [15] M. W. WIBISONO, A. P. KURNIATI, G. A. A. WISUDIAWAN: *Process Mining using Inductive Miner Algorithm to Determine the actual Business Process Model*, JURIKOM (Jurnal Riset Komputer) (2022), DOI: [10.30865/jurikom.v9i4.4769](https://doi.org/10.30865/jurikom.v9i4.4769).
- [16] W. ZAITSU, M. JIN: *Distinguishing ChatGPT(-3.5, -4)-generated and human-written papers through Japanese stylometric analysis*, PLOS ONE 18.8 (2023), e0288453, DOI: [10.1371/journal.pone.0288453](https://doi.org/10.1371/journal.pone.0288453).

ChatPULI: Enhancement to the first Hungarian conversational model

Zijian Győző Yang, Ágnes Bánfi, Réka Dodé,
Gergő Ferenczi, Flóra Földesi, Péter Hatvani, Enikő Héja,
Mariann Lengyel, Gábor Madarász, Mátyás Osváth,
Bence Sárossy, Kristóf Varga, Tamás Váradi,
Gábor Prószéky, Noémi Ligeti-Nagy

ELTE Research Centre for Linguistics

[yang.zijian.gyozo, banfi.agnes, dode.reka ferenczi.gergo foldesi.flora hatvani.peter
heja.eniko lengyel.mariann madarasz.gabor osvath.matyas sarossy.bence varga.kristof
varadi.tamas proszeky.gabor ligeti-nagy.noemi][@nytud.elte.hu](mailto:nytud.elte.hu)

Abstract. This paper presents the development and evaluation of PULI-LlumiX-Llama-3.1 Chat and PULI Trio Q Chat, the first Hungarian-focused conversational large language models based on the Llama 3.1 and Qwen 2.5 architectures. Extending previous work on Hungarian instruction-following models, we applied continual pre-training on multilingual and Hungarian corpora, followed by supervised fine-tuning on an expanded instruction dataset including Hungarian, English, and Chinese prompts. Our models demonstrate significant performance improvements on Hungarian language understanding benchmarks, as well as on few-shot and zero-shot tasks, compared to earlier PULI models. Additionally, they show enhanced capabilities in machine translation and multi-turn dialogue handling. These results highlight the effectiveness of continual pre-training and fine-tuning strategies for adapting large language models to low-resource languages like Hungarian, and provide a foundation for future research in conversational AI for under-represented languages.

Keywords: PULI models, Llama, Qwen, large language model, conversational language model

AMS Subject Classification: 68T07, 68T50, 68U15

1. Background

In recent years, large language models (LLMs) have gained significant attention, as companies compete to develop models capable of solving a wide range of natural language processing tasks through extensive data training. The release of ChatGPT¹ by OpenAI² showcased unprecedented capabilities, achieved through a multi-stage fine-tuning process. Nevertheless, while ChatGPT and similar large language models have been extensively trained for major languages, no conversational model has been developed specifically for Hungarian. Companies such as OpenAI typically include minimal Hungarian content in their training corpora, as is the case with most small, low-resource language with limited amount of available datasets.

For Hungarian, the first instruction-following model was introduced by Yang et al. [34], based on the Llama 2 architecture [29]. While this model demonstrated significantly improved performance compared to previous Hungarian LLMs on local benchmarks, it still lacks key functionality, most notably, the ability to engage in natural, conversational interaction.

Conversational fine-tuning datasets require substantial human resources due to their multi-turn interaction structure, and, as mentioned earlier, such datasets are currently lacking in the Hungarian language. In our recent research, we leveraged the advantages of transfer learning and, following the approach of Yang et al. [34], we applied continual pre-training to the Llama 3.1 8B Instruct [10] and Qwen2.5 7B Instruct [31] models to adapt their conversational capabilities to the Hungarian language. Our newly continual pre-trained base models are named **PULI-LlumiX-Llama-3.1** and **PULI Trio Q**. The name “PULI” refers to the eponymous Hungarian dog breed. This dog is not only a cultural symbol in Hungary, but is also known for its agility, energy, and intelligence despite its small size. By choosing this name, we emphasize similar qualities in our small but capable model. Our base models are available at our Hugging Face repository³.

2. Related work

The global prominence for decoder-only transformer-based large language chat models was brought forth by OpenAI’s GPT model family [21] which was primarily trained on English data. In the past years, an array of new models have appeared at a rapid pace. Besides OpenAI’s GPT series, the most notable closed-source families include Claude [2] by Anthropic and Gemini [7] by Google. More recently, open-source Deepseek models [8] have emerged as one of the most notable developments within the LLM landscape. They introduced techniques such as supervised fine-tuning (SFT) and Group Relative Policy Optimization (GRPO),

¹<https://chatgpt.com>

²<https://openai.com>

³<https://huggingface.co/NYTK>

which helped improving training efficiency and lowering computational costs, while maintaining competitive performance.

Other open-source model families include Alibaba’s Qwen series and the subsequent Qwen2 models [31], which comprise a range of base models as well as instruction-tuned LLMs, spanning from 0.5B to 72B parameters. The series also includes a Mixture-of-Experts (MoE) model, further broadening the available architectures and training strategies within open-source development.

In some instances makers of closed-source model families also release open-source series such as is the case with Google’s Gemma model family. Their latest model, Gemma 3 [28] is a multimodal and multilingual model, available in various parameter sizes ranging from 1B to 27B, with a 128k-token context window.

Meta’s Llama series [29, 30] has also played a central role in improving access to high-performance models. Their multilingual pre-training, range of model sizes, and general-purpose capabilities have made them widely adopted, especially in contexts of smaller language model development, where training data and infrastructural capacities are limited.

Llama-2-7b was adapted to Hungarian by SambaLingo for its pre-trained bilingual base model, Sambalingo-Hungarian-Base [6]. The model, that also has its chat version, was trained on 59 billion tokens sourced from the Hungarian split of the Cultura-X corpus [19]. While it represents a significant step for Hungarian LLM-development, the monolithic nature of the training data places limitations on its ability to recognize the nuances and genre diversity of the language.

Finally, another model family with notable Hungarian representation is the EuroLLM series [18]. As of the time of writing, their flagship model, EuroLLM-9B, was trained on 4 trillion tokens drawn from all EU official languages. However, during pretraining – prior to instruction tuning and alignment – the dataset remained predominantly English, with the language comprising 32.5–50% of the overall token mix. This raises concerns about the effective representation of mid- and low-resource languages in the model, including Hungarian.

Due to limited computational resources, we opted for an approach involving continual pre-training. This method allows us to incrementally update the model with new data without retraining from scratch, making it a more resource-efficient alternative.

3. Datasets and PULI models

In our continual pre-training experiment, we followed the methodology of [34]. In Table 1, we present the main characteristics of the continual pre-training corpora. For the PULI long and English parts, we used the same corpora (containing 8,077,464,716 words) as employed in the work of Yang et al. [34], but we additionally incorporated the Hungarian Wikipedia to enhance the model’s knowledge of the Hungarian language. For the Chinese part, we used the Wu Dao 2.0 corpus [37], which consists of long documents (over 15,000 characters) drawn from its publicly available portion. During the pre-training experiments, we observed that

the model generated more grammaticality errors in its outputs. To address this issue, we concluded the training with a Hungarian-only dataset. In this final phase, our goal was not only to enhance the model’s proficiency in Hungarian, but also to provide it with up-to-date information. Accordingly, the dataset was composed of carefully selected Hungarian articles published over the past six years. The corpora used in this stage were: Hungarian Wikipedia (2019–2025), containing 174,945,601 words, and Hungarian news articles (2019–2025), containing 622,275,769 words. The characteristics of the final-stage corpus are presented in Table 1.

Table 1. Corpus characteristics for continual pre-training.

	Documents	Words	Average document length average / median (word)
PULI long (hu)	763 704	7 902 519 115	10 823.38 / 7 149
Long Context QA (en)	88 957	1 009 562 704	11 348.88 / 11 274
BookSum (en)	9 600	42 339 698	4 410.39 / 3 266
Wu Dao 2.0 long (zh)	174 118	2 855 217 266 (Characters)	16 398.17 / 14 767 (Characters)
Final stage (hu)	1 411 979	797 221 370	443.65 / 217

The continual pre-training phase employed different multilingual mixtures for each model. The PULI-LlumiX-Llama-3.1 (bilingual) model was trained on a combination of the PULI long and English components. In contrast, the PULI Trio Q (trilingual) model utilized a mixture of the PULI long, English, and Chinese components. The training process for both models concluded with a final stage using exclusively Hungarian data.

For fine-tuning, we used the same basic corpus as in the work of Yang et al. [34], but we significantly expanded the instruction dataset. To better align the model with conversational use, we also collected a large number of dialogue-style prompts and replaced the Stanford Alpaca instruction template with the Llama 3 and Qwen2 chat templates to ensure consistency with the base models’ instruction style. Our contributions to the fine-tuning dataset include:

- HuLU benchmark [15] prompts: Increased to 7,344 segments;
- MILQA benchmark [20] prompts: Increased to 2,191 segments;
- Graduation tasks: Increased to 5,589 segments;
- Title, keyword generation, and summarization tasks [3, 33]: Increased to 3,000 segments;
- Conversational and instructional prompts [14, 17, 26, 38]: Hungarian-specific content, totaling 10,884 prompts;
- Psychiatric content [9]: 315 interviews;

- Text simplification [24]: 2,274 prompts;
- Named Entity Recognition prompts: 2,300 prompts generated from NYTK-NerKor [25];
- University exam: 166 publicly available questions covering tourism, business, geography, and economics.
- User questions: Increased to 935 segments. These prompts are generated by questions that users ask from our demo models. In these cases, the answers are manually written by annotators.
- Poem prompts: 11,800 prompts collected from Hugging Face⁴.
- Miscellaneous: Increased to 3,358 segments. Prompts from different customer datasets that can be made public.

In total, our Hungarian fine-tuning dataset consists of 44,626 segments. In our supervised fine-tuning, inspired by Yang et al. [34], we mixed English and Chinese prompts to our Hungarian prompts:

- Bilingual prompt dataset: In the case of Llumix model, we added 85,775 English prompts: Stanford Alpaca [27], Dolly [5] and llama-instruct⁵.
- Trilingual prompt dataset: In the case of Trio model, beside the English prompts, we added 50,130 Chinese prompts: neo_sft_phase2⁶ Chinese part.

To evaluate the performance of our models, we compared them with the previous PULI models:

- **PULI 3SX** is a GPT-NeoX-based [4] model with 6.85 billion parameters, pre-trained from scratch on a monolingual Hungarian corpus of 36.3 billion words.
- **PULI Trio**: Another GPT-NeoX-based model, with 7.67 billion parameters, trained from scratch as a Hungarian-English-Chinese trilingual model. The Hungarian subset of the training data contains 41.5 billion words.
- **PULI Llumix**: A continued pre-training adaptation of Llama 2 [30]. The original Llama 2 model was trained on approximately 600 million Hungarian tokens. This version was further trained on an additional 7.9 billion Hungarian words, emphasizing long-form documents exceeding 5,000 words. Additionally, the context window has been expanded to 32,768 tokens.

⁴<https://huggingface.co/datasets/leinadsened/hungarian-poems-with-instructions>

⁵<https://huggingface.co/datasets/togethercomputer/llama-instruct>

⁶https://huggingface.co/datasets/m-a-p/neo_sft_phase2

4. Methods and experiments

In our current research, we adapted two multilingual LLMs to Hungarian. Our goal was to balance computational efficiency and performance; therefore, inspired by [34], we selected the Llama 3.1 8B Instruct [10] and Qwen2.5 7B Instruct for continued pre-training to adapt it to Hungarian. According to prior research, transfer learning plays a crucial role in language adaptation, and instruction-tuned models contain valuable structured knowledge compared to base models. However, we found no previous research that applies continued pre-training on an instruct model for language adaptation.

For the training, we used the LLaMA-Factory [40] framework with the following hyperparameters: global batch size = 128; learning rate = 1e-5; precision = bf16 (Llama) / fp32 (Qwen); context length = 16,384 (Llama) and 32,768 (Qwen); epochs = 1.

We trained two base models from the Llama and Qwen Instruct models, and two conversational or chat models from it:

- **PULI-LlumiX-Llama-3.1:** The Llama 3.1 8B Instruct model were continual pre-trained on bilingual (Hungarian-English) dataset, then completed on monolingual Hungarian dataset.
- **PULI-LlumiX-Llama-3.1 Chat:** The PULI-LlumiX-Llama-3.1 model were supervised fine-tuned on our bilingual (Hungarian-English) fine-tuning dataset, with 3 epochs.
- **PULI Trio Q:** The Qwen2.5 7B Instruct model were continual pre-trained on trilingual (Hungarian-English-Chinese) dataset, then completed on monolingual Hungarian dataset.
- **PULI Trio Q Chat:** The PULI Trio Q model were supervised fine-tuned on our trilingual (Hungarian-English-Chinese) fine-tuning dataset, with 3 epochs.

For the conversational supervised fine-tuning, we also used the LLaMA-Factory framework with the following hyperparameters: global batch size = 64; learning rate = 1e-5; precision = bf16 (Llama) / fp32 (Qwen); context length = 32,768; epochs = 3.

For the fine-tuning evaluation on the HuLU benchmarks, we used the Hugging Face implementation⁷.

All training was conducted on four NVIDIA A100 GPUs (80 GB each), utilizing the Accelerate library [11], Fully Sharded Data Parallel [39] and Liger Kernel [12] in all cases.

For the few-shot and zero-shot evaluation tasks, to ensure a fair comparison, we applied the same settings, evaluation metrics, prompts, and methodologies as described in the study by Yang et al. [35].

⁷<https://github.com/huggingface/transformers/tree/main/examples/pytorch>

5. Results and evaluation

First, we examined the vocabularies and the Hungarian training data subsets of the openly available Hungarian LLMs. In Table 2, the Hungarian dataset/subset (in words), vocabulary sizes, and fertility values [1] are observed. The fertility values were calculated from the largest PULI corpus [36], which consists of 41 billion words. Table 2 highlights the impact of training corpus composition on vocabulary and fertility values. The earlier PULI models (3SX and Trio), trained exclusively or primarily on Hungarian data, used vocabularies optimized for Hungarian morphology and word formation, resulting in lower fertility values (1.58–1.63). This reflects a tighter, language-specific tokenization. In contrast, the newer LlumiX and Trio Q models – pre-trained on multilingual corpora with larger token vocabularies – exhibit higher fertility (around 3.0), indicating a shift toward tokenization schemes influenced by multilingual instruction models. This change reflects a balance between leveraging multilingual model architectures and retaining adequate representation for Hungarian.

Table 2. Hungarian datasets, vocabularies and fertilities of PULI LLMs.

	Dataset	Vocab	Fertility
PULI 3SX	36.3B	50,048	1.634
PULI Trio	41.5B	150,016	1.585
PULI LlumiX	7.9B	32,000	3.058
PULI-LlumiX-Llama-3.1	9.7B	128,256	3.062
PULI Trio Q	9.7B	152,064	3.126

Table 3. Few-shot and zero-shot results.

	Few-shot		
	HuCOLA	HuRTE	HuSST
PULI 3SX	51.51	52.40	54.57
PULI Trio	53.47	54.30	54.96
PULI LlumiX	55.84	61.49	68.53
PULI-LlumiX-Llama-3.1	71.26	72.63	69.38
PULI Trio Q	63.85	62.98	70.79
	Zero-shot		
PULI 3SX Instruct	61.76	52.09	46.27
PULI Trio Instruct	51.97	74.54	54.50
PULI LlumiX Instruct	66.98	74.54	70.06
PULI-LlumiX-Llama-3.1 Chat	72.98	86.76	79.44
PULI Trio Q Chat	76.68	86.10	78.04

In our second evaluation phase, we conducted few-shot and zero-shot experiments. To ensure a fair comparison with the previous work of Yang et al. [35], we evaluated our models on HuCOLA, HuRTE, and HuSST using the same methodologies and metrics (balanced accuracy). Performance improved consistently across both few-shot and zero-shot tasks from PULI 3SX to the latest models. The PULI-LlumiX-Llama-3.1 Chat and PULI Trio Q models outperformed all previous iterations, with PULI-LlumiX-Llama-3.1 Chat achieving the highest scores on the HuCOLA, HuRTE, and HuSST benchmarks compared to PULI Trio Q. This underscores the effectiveness of continual pre-training and conversational fine-tuning on Hungarian datasets in enhancing model generalization. Interestingly, in the HuCOLA task, the Qwen-based PULI Trio Q model achieved the best result, demonstrating strong adaptability to specific tasks, even when those tasks are linguistically specialized.

Table 4 presents the few-shot and zero-shot experiments on English–Hungarian (en–hu) and Hungarian–English (hu–en) machine translation tasks. The evaluation metrics used were SacreBLEU [23] and ChrF [22], reported as BLEU / ChrF scores. The results show that the latest models exhibit significantly improved translation capabilities compared to the earlier PULI versions. In both few-shot and zero-shot translation tasks, the PULI-LlumiX-Llama-3.1 Chat and PULI Trio Q Chat models displayed marked improvements over their predecessors. The Chat variants significantly outperformed the earlier LlumiX and Trio models, particularly in the zero-shot setting, with PULI-LlumiX-Llama-3.1 Chat achieving the highest BLEU and ChrF scores. This suggests that conversational fine-tuning not only enhanced dialogue handling but also contributed to stronger cross-lingual transfer and translation capabilities.

Table 4. Comparison of PULI multilingual models on machine translation task.

	en-hu	hu-en
	Few-shot	
PULI Trio	14.42 / 45.12	19.69 / 48.27
PULI LlumiX	13.73 / 42.30	15.43 / 35.84
PULI-LlumiX-Llama-3.1	19.31 / 50.58	29.27 / 56.77
PULI Trio Q	21.24 / 52.68	31.42 / 59.41
	Zero-shot	
PULI Trio Instruct	24.86 / 52.51	26.88 / 53.34
PULI LlumiX Instruct	18.32 / 50.63	27.85 / 56.56
PULI-LlumiX-Llama-3.1 Chat	37.79 / 63.02	43.62 / 66.79
PULI Trio Q Chat	30.08 / 58.39	38.45 / 63.58

In the third phase of our evaluation, we performed fine-tuning experiments on our LLMs. Table 5 presents the fine-tuning performance of our new models. For evaluation, we further fine-tuned the models on four HuLU benchmarks [15] using

the LoRA [13] and full parameter fine-tuning method. The evaluation metrics were the original HuLU metrics, Matthews Correlation Coefficient (MCC) and Accuracy (ACC), respectively. We experimented with various hyperparameters; in the results tables, we report only the best results achieved. The results from HuLU benchmarks reinforce the superior performance of the conversationally fine-tuned models. PULI-LlumiX-Llama-3.1 Chat consistently achieved the best scores across all tasks, notably surpassing previous models in HuCOLA, HuCoPA, HuRTE, and HuSST. PULI Trio Q Chat, while trailing slightly behind LlumiX, still outperformed all older PULI models. This demonstrates the clear advantage of fine-tuning with a carefully designed conversational dataset, especially when paired with continual pre-training.

Table 5. Performance of the PULI-LlumiX-Llama-3.1 and PULI Trio Q models, on the HuLU benchmarks.

	HuCOLA (MCC)	HuCoPA (MCC)	HuRTE (MCC)	HuSST (ACC)
PULI 3SX Instruct	59.5	5.3	55.5	79.7
PULI Trio Instruct	63.1	44.5	59.2	79.1
PULI LlumiX Instruct	70.3	73.4	68.2	81.9
PULI-LlumiX-Llama-3.1 Base	69.2	74.1	71.9	82.2
PULI-LlumiX-Llama-3.1 Chat	72.54	74.1	76.0	83.1
PULI Trio Q Chat	64.7	67.1	68.8	82.1

We also conducted an experiment in which we fine-tuned the PULI-LlumiX-Llama-3.1 Base and PULI-LlumiX-Llama-3.1 Chat models. As expected, the Chat model achieved higher results in this comparison, likely because it had already encountered some examples from the HuLU benchmarks during its conversational training.

A sample conversation is shown in Table 6, demonstrating the model’s (PULI-LlumiX-Llama-3.1 Chat) ability to follow instructions and maintain context across multiple turns. In this example, we asked the model to list three characteristics for various entities: dog, snail, table, carrot, love, and the transcendent. At the end of the conversation, we asked it to list four characteristics instead. As shown in Table 6, the model consistently provided three relevant features for each entity, and correctly responded with four when requested.

6. Conclusion

In this work, we introduce **PULI-LlumiX-Llama-3.1 Chat** and **PULI Trio Q Chat**, our first conversational Hungarian large language models, based on the Llama 3.1 and Qwen2 architectures. Building upon the foundation laid by Yang et al. [34], we expanded both the pre-training and fine-tuning datasets. Our new model achieved notable improvements across multiple HuLU benchmarks and

demonstrated enhanced conversational capabilities for Hungarian compared to previous Llumix versions.

Table 6. Example snippet of a conversation.

User:	mondj három tulajdonságot a kutyákról (‘tell me three characteristics about dogs.’)
Assistant:	1. A kutyák hűségesek és odaadóak gazdáik iránt. 2. A kutyák képesek felismerni és reagálni az emberi érzelmekre. 3. A kutyák egyszerű képességgel rendelkeznek a szagokat és zajokat érzékelni.
User:	csigák? (‘snails?’)
Assistant:	1. A csigák ügyesek a mászásban és kiválóan képesek felmászni a felületeken. 2. A csigák képesek kemény héjú otthonukat önmagukban visszahúzni és védelmet nyújtani számukra. 3. A csigák nagyon válogatósak és csak bizonyos növényeket táplálkoznak, melyek létfontosságú tápanyagokat biztosítanak számukra.
...	...
User:	és a transzcendens? (‘and the transcendent?’)
Assistant:	1. A transzcendens olyan fogalom, amely a mindennapi élettől való fizikai vagy szellemi távolságra utal, és gyakran kapcsolódik a spirituális vagy misztikus tapasztalatokhoz. 2. A transzcendens élmények olyan intenzív és mély élmények, amelyek túlmutatnak az érzékelhető világon, és gyakran egyfajta megvilágosodással, békeélménnyel vagy egyesülési élménnyel járnak. 3. A transzcendens élmények gyakran meghatározó és megváltoztató hatással lehetnek az emberek életére, inspirálva őket a személyes fejlődésre vagy a vallási gyakorlatokba való mélyebb elmerülésre.
User:	legyen négy (‘let there be four’)
Assistant:	1. A transzcendens olyan fogalom, amely a mindennapi élettől való fizikai vagy szellemi távolságra utal, és gyakran kapcsolódik a spirituális vagy misztikus tapasztalatokhoz. 2. A transzcendens élmények olyan intenzív és mély élmények, amelyek túlmutatnak az érzékelhető világon, és gyakran egyfajta megvilágosodással, békeélménnyel vagy egyesülési élménnyel járnak. 3. A transzcendens élmények gyakran meghatározó és megváltoztató hatással lehetnek az emberek életére, inspirálva őket a személyes fejlődésre vagy a vallási gyakorlatokba való mélyebb elmerülésre. 4. A transzcendens több mint a szó szoros értelmében vett élmény lehet, hanem egyfajta tudatállapot vagy spirituális tudatosság, mely gyakran meditáció vagy transzcendentális gyakorlatok segítségével érhető el.

These results also contribute to a possible framework of developing conversational models for small languages through transfer learning and the application of

continual pre-praining methods. Still, it is worth noting that while our research delivered measurable improvements, the results were limited by the need for manual dataset curation, which remains a resource-intensive and time-consuming process.

In the future, we plan to evaluate our conversational models using LLM evaluation benchmarks, as two new benchmarks for Hungarian, HuGME [16] and Open-HuEval [32], have been released in recent weeks.

References

- [1] J. ÁCS: *Exploring bert’s vocabulary*, 2019, URL: <https://juditacs.github.io/2019/02/19/bert-tokenization-stats.html> (visited on 02/12/2025).
- [2] ANTHROPIC: *Claude 3: Haiku, Sonnet, and Opus — the next generation of Claude models*, blog post / online announcement, Available at <https://www.anthropic.com/research/claude-3-family>, 2024.
- [3] B. BARTA, D. LAKATOS, A. NAGY, M. K. NYIST, J. ÁCS: *HunSum-1: an Abstractive Summarization Dataset for Hungarian*, in: XIX. Hungarian Computational Linguistics Conference, Szeged, Hungary: University of Szeged, 2023, pp. 231–243.
- [4] S. BLACK, S. BIDERMAN, E. HALLAHAN, Q. ANTHONY, L. GAO, L. GOLDING, H. HE, C. LEAHY, K. McDONELL, J. PHANG, M. PIELER, U. S. PRASHANTH, S. PUROHIT, L. REYNOLDS, J. TOW, B. WANG, S. WEINBACH: *GPT-NeoX-20B: An Open-Source Autoregressive Language Model*, in: Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models, ed. by A. FAN, S. ILIC, T. WOLF, M. GALLÉ, virtual+Dublin: Association for Computational Linguistics, May 2022, pp. 95–136, DOI: [10.18653/v1/2022.bigscience-1.9](https://doi.org/10.18653/v1/2022.bigscience-1.9), URL: <https://aclanthology.org/2022.bigscience-1.9/>.
- [5] M. CONOVER, M. HAYES, A. MATHUR, J. XIE, J. WAN, S. SHAH, A. GHODSI, P. WENDELL, M. ZAHARIA, R. XIN: *Free Dolly: Introducing the World’s First Truly Open Instruction-Tuned LLM*, 2023, URL: <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm> (visited on 06/30/2023).
- [6] Z. CSAKI, B. LI, J. L. LI, Q. XU, P. PAWAKAPAN, L. ZHANG, Y. DU, H. ZHAO, C. HU, U. THAKKER: *SambaLingo: Teaching Large Language Models New Languages*, in: Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024), ed. by J. SÄLEVÄ, A. OWODUNNI, Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 1–21, DOI: [10.18653/v1/2024.mrl-1.1](https://doi.org/10.18653/v1/2024.mrl-1.1), URL: <https://aclanthology.org/2024.mrl-1.1/>.
- [7] G. DEEPMIND: *Introducing Gemini: our most capable and general model*, <https://blog.google/technology/ai/google-gemini-ai/>, Accessed: 2025-09-29, 2023.
- [8] DEEPSEEK-AI: *DeepSeek LLM: Scaling Open-Source Language Models with Longtermism*, arXiv preprint arXiv:2401.02954 (2024), URL: <https://github.com/deepseek-ai/DeepSeek-LLM>.
- [9] F. FELLETÁR, G. GOSZTOLYA, Z. G. YANG, I. HOFFMANN, A. BABARCZY, Z. S. UNOKA: *Associations between coherence and temporal parameters of narrative speech production in borderline personality disorder*, Journal of Psychiatric Research 188 (2025), pp. 218–228, ISSN: 0022-3956, DOI: [10.1016/j.jpsychires.2025.05.063](https://doi.org/10.1016/j.jpsychires.2025.05.063), URL: <https://www.sciencedirect.com/science/article/pii/S0022395625003632>.
- [10] A. GRATTAFIORI ET AL.: *The Llama 3 Herd of Models*, 2024, arXiv: [2407.21783 \[cs.AI\]](https://arxiv.org/abs/2407.21783), URL: <https://arxiv.org/abs/2407.21783>.
- [11] S. GUGGER, L. DEBUT, T. WOLF, P. SCHMID, Z. MUELLER, S. MANGRULKAR, M. SUN, B. BOSSAN: *Accelerate: Training and inference at scale made simple, efficient and adaptable*. <https://github.com/huggingface/accelerate>, 2022.

- [12] P.-L. HSU, Y. DAI, V. KOTHAPALLI, Q. SONG, S. TANG, S. ZHU, S. SHIMIZU, S. SAHNI, H. NING, Y. CHEN, Z. WANG: *Liger-Kernel: Efficient Triton Kernels for LLM Training*, in: *Championing Open-source DEvelopment in ML Workshop @ ICML25*, 2025, URL: <https://openreview.net/forum?id=36SjAIT42G>.
- [13] E. J. HU, yelong SHEN, P. WALLIS, Z. ALLEN-ZHU, Y. LI, S. WANG, L. WANG, W. CHEN: *LoRA: Low-Rank Adaptation of Large Language Models*, in: *International Conference on Learning Representations*, 2022, URL: <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [14] A. KÖPF, Y. KILCHER, D. VON RÜTTE, S. ANAGNOSTIDIS, Z.-R. TAM, K. STEVENS, A. BARHOUM, N. M. DUC, O. STANLEY, R. NAGYFI, S. ES, S. SURI, D. GLUSHKOV, A. DANTULURI, A. MAGUIRE, C. SCHUHMAN, H. NGUYEN, A. MATTICK: *OpenAssistant conversations - democratizing large language model alignment*, in: *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, New Orleans, LA, USA: Curran Associates Inc., 2023.
- [15] N. LIGETI-NAGY, G. FERENCZI, E. HÉJA, L. J. LAKI, N. VADÁSZ, Z. G. YANG, T. VÁRADI: *HuLU: Hungarian Language Understanding Benchmark Kit*, in: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, ed. by N. CALZOLARI, M.-Y. KAN, V. HOSTE, A. LENCI, S. SAKTI, N. XUE, Torino, Italia: ELRA and ICCL, May 2024, pp. 8360–8371, URL: <https://aclanthology.org/2024.lrec-main.733/>.
- [16] N. LIGETI-NAGY, G. MADARASZ, F. FOLDESI, M. LENGUEL, M. OSVATH, B. SAROSSY, K. VARGA, G. Z. YANG, E. HÉJA, T. VÁRADI, G. PRÓSZÉKY: *HuGME: A benchmark system for evaluating Hungarian generative LLMs*, in: *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM²)*, ed. by O. ARVIV, M. CLINCIU, K. Dhole, R. DROR, S. GEHRMANN, E. HABBA, I. ITZHAK, S. MILLE, Y. PERLITZ, E. SANTUS, J. SEDOC, M. SHMUELI SCHEUER, G. STANOVSKY, O. TAFJORD, Vienna, Austria and virtual meeting: Association for Computational Linguistics, July 2025, pp. 385–403, ISBN: 979-8-89176-261-9, URL: <https://aclanthology.org/2025.gem-1.32/>.
- [17] S. LONGPRE, Y. LU, J. DAIBER: *MKQA: A Linguistically Diverse Benchmark for Multilingual Open Domain Question Answering*, *Transactions of the Association for Computational Linguistics* 9 (2021), ed. by B. ROARK, A. NENKOVA, pp. 1389–1406, DOI: [10.1162/tacl_a_00433](https://doi.org/10.1162/tacl_a_00433), URL: <https://aclanthology.org/2021.tacl-1.82/>.
- [18] P. H. MARTINS, P. FERNANDES, J. ALVES, N. M. GUERREIRO, R. REI, D. M. ALVES, J. POMBAL, A. FARAJIAN, M. FAYSSE, M. KLIMASZEWSKI, P. COLOMBO, B. HADDOW, J. G. C. DE SOUZA, A. BIRCH, A. F. T. MARTINS: *EuroLLM: Multilingual Language Models for Europe*, 2024, arXiv: [2409.16235](https://arxiv.org/abs/2409.16235) [cs.CL], URL: <https://arxiv.org/abs/2409.16235>.
- [19] T. NGUYEN, C. V. NGUYEN, V. D. LAI, H. MAN, N. T. NGO, F. DERNONCOURT, R. A. ROSSI, T. H. NGUYEN: *CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages*, in: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, ed. by N. CALZOLARI, M.-Y. KAN, V. HOSTE, A. LENCI, S. SAKTI, N. XUE, Torino, Italia: ELRA and ICCL, May 2024, pp. 4226–4237, URL: <https://aclanthology.org/2024.lrec-main.377>.
- [20] A. NOVÁK, B. NOVÁK, T. ZOMBORI, G. SZABÓ, Z. SZÁNTÓ, R. FARKAS: *A Question Answering Benchmark Database for Hungarian*, in: *Proceedings of the 17th Linguistic Annotation Workshop (LAW-XVII)*, ed. by J. PRANGE, A. FRIEDRICH, Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 188–198, DOI: [10.18653/v1/2023.law-1.19](https://doi.org/10.18653/v1/2023.law-1.19), URL: <https://aclanthology.org/2023.law-1.19/>.
- [21] OPENAI: *Introducing ChatGPT*, <https://openai.com/blog/chatgpt>, Accessed: 2025-07-24, 2022.
- [22] M. POPOVIĆ: *chrF: character n-gram F-score for automatic MT evaluation*, in: *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 392–395, DOI: [10.18653/v1/W15-3049](https://doi.org/10.18653/v1/W15-3049), URL: <https://aclanthology.org/W15-3049>.

- [23] M. POST: *A Call for Clarity in Reporting BLEU Scores*, in: Proceedings of the Third Conference on Machine Translation: Research Papers, Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 186–191, URL: <https://www.aclweb.org/anthology/W18-6319>.
- [24] N. PRÓTÁR, D. M. NEMESKEY: *HunSimpleNews: Az első autentikus magyar nyelvű szövegek-ből álló szövegegyszerűsítési korpusz [The first text simplification corpus consisting of authentic Hungarian-language texts]*, in: XXI. Hungarian Computational Linguistics Conference, Szeged, Hungary, 2025, pp. 197–218.
- [25] E. SIMON, N. VADÁSZ: *Introducing NYTK-NerKor, A Gold Standard Hungarian Named Entity Annotated Corpus*, in: Text, Speech, and Dialogue - 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6-9, 2021, Proceedings, ed. by K. EKSTEIN, F. PÁRTL, M. KONOPÍK, vol. 12848, Lecture Notes in Computer Science, Springer, 2021, pp. 222–234, DOI: [10.1007/978-3-030-83527-9_19](https://doi.org/10.1007/978-3-030-83527-9_19).
- [26] S. SINGH, F. VARGUS, D. D’SOUZA, B. KARLSSON, A. MAHENDIRAN, W.-Y. KO, H. SHANDILYA, J. PATEL, D. MATACIUNAS, L. O’MAHONY, M. ZHANG, R. HETTIARACHCHI, J. WILSON, M. MACHADO, L. MOURA, D. KRZEMIŃSKI, H. FADAEI, I. ERGUN, I. OKOH, A. ALAAGIB, O. MUDANNAYAKE, Z. ALYAFEAI, V. CHIEN, S. RUDER, S. GUTHIKONDA, E. ALGHAMDI, S. GEHRMANN, N. MUENNIGHOFF, M. BARTOLO, J. KREUTZER, A. ÜSTÜN, M. FADAEI, S. HOOKER: *Aya Dataset: An Open-Access Collection for Multilingual Instruction Tuning*, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ed. by L.-W. KU, A. MARTINS, V. SRIKUMAR, Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 11521–11567, DOI: [10.18653/v1/2024.acl-long.620](https://doi.org/10.18653/v1/2024.acl-long.620), URL: <https://aclanthology.org/2024.acl-long.620/>.
- [27] R. TAORI, I. GULRAJANI, T. ZHANG, Y. DUBOIS, X. LI, C. GUESTRIN, P. LIANG, T. B. HASHIMOTO: *Stanford Alpaca: An Instruction-following LLaMA model*, 2023, URL: [5Curl%7Bhttps://github.com/tatsu-lab/stanford%5C_alpaca%7D](https://github.com/tatsu-lab/stanford%5C_alpaca%7D).
- [28] G. TEAM: *Gemma 3* (2025), URL: <https://arxiv.org/abs/2503.19786>.
- [29] H. TOUVRON ET AL.: *Llama 2: Open Foundation and Fine-Tuned Chat Models* (2023), arXiv: [2307.09288](https://arxiv.org/abs/2307.09288) [cs.CL].
- [30] H. TOUVRON ET AL.: *Llama 2: Open Foundation and Fine-Tuned Chat Models*, 2023, arXiv: [2307.09288](https://arxiv.org/abs/2307.09288) [cs.CL], URL: <https://arxiv.org/abs/2307.09288>.
- [31] A. YANG ET AL.: *Qwen2 Technical Report*, arXiv preprint arXiv:2407.10671 (2024).
- [32] H. YANG, X. WEI, J. WU, N. LIGETI-NAGY, J. SUN, Y. WANG, G. Z. YANG, J. GAO, J. WANG, B. JIANG, S. WANG, N. YU, Z. ZHANG, S. HONG, H. LIU, W. LI, S. ZHANG, D. LIN, L. WU, G. PRÓSZÉKY, C. HE: *OpenHuEval: Evaluating Large Language Model on Hungarian Specifics*, in: Findings of the Association for Computational Linguistics: ACL 2025, ed. by W. CHE, J. NABENDE, E. SHUTOVA, M. T. PILEHVAR, Vienna, Austria: Association for Computational Linguistics, July 2025, pp. 7464–7520, ISBN: 979-8-89176-256-5, URL: <https://aclanthology.org/2025.findings-acl.390/>.
- [33] Z. G. YANG: *Neural text summarization for Hungarian*, Acta Linguistica Academica 69.4 (2022), pp. 474–500, DOI: [10.1556/2062.2022.00577](https://doi.org/10.1556/2062.2022.00577), URL: <https://akjournals.com/view/journals/2062/69/4/article-p474.xml>.
- [34] Z. G. YANG, R. DODÉ, G. FERENCZI, P. HATVANI, E. HÉJA, G. MADARÁSZ, N. LIGETI-NAGY, B. SÁROSSY, Z. SZANISZLÓ, T. VÁRADI, T. VEREBÉLYI, G. PRÓSZÉKY: *The First Instruction-Following Large Language Models for Hungarian*, in: 2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS) Proceedings, Debrecen, Hungary: University of Debrecen, 2024, pp. 247–252, ISBN: 9798350387889.
- [35] Z. G. YANG, R. DODÉ, G. FERENCZI, H. PÉTER, E. HÉJA, M. LENGYEL, N. LIGETI-NAGY, G. MADARÁSZ, B. SÁROSSY, K. VARGA, T. VARGA, T. VÁRADI: *PULI LlmüX modell - Egy folytatólagosan előtanított nagy nyelvi modell [PULI LlmüX Model – A Continual Pre-Trained Large Language Model]*, in: XXI. Hungarian Computational Linguistics Conference, Szeged, Hungary: University of Szeged, 2025, pp. 153–167.

- [36] Z. G. YANG, L. J. LAKI, T. VÁRADI, G. PRÓSZÉKY: *Mono- and multilingual GPT-3 models for Hungarian*, in: Text, Speech, and Dialogue, Lecture Notes in Computer Science, Plzeň, Czech Republic: Springer Nature Switzerland, 2023, pp. 94–104, ISBN: 978-3-031-40498-6.
- [37] S. YUAN, H. ZHAO, Z. DU, M. DING, X. LIU, Y. CEN, X. ZOU, Z. YANG, J. TANG: *WuDao-Corpora: A super large-scale Chinese corpora for pre-training language models*, AI Open 2 (2021), pp. 65–68, ISSN: 2666-6510, DOI: [10.1016/j.aiopen.2021.06.001](https://doi.org/10.1016/j.aiopen.2021.06.001).
- [38] W. ZHAO, X. REN, J. HESSEL, C. CARDIE, Y. CHOI, Y. DENG: *WildChat: 1M ChatGPT Interaction Logs in the Wild*, in: The Twelfth International Conference on Learning Representations, 2024, URL: <https://openreview.net/forum?id=Bl8u7ZR1bM>.
- [39] Y. ZHAO, A. GU, R. VARMA, L. LUO, C.-C. HUANG, M. XU, L. WRIGHT, H. SHOJANAZERI, M. OTT, S. SHLEIFER, A. DESMAISON, C. BALIOGLU, P. DAMANIA, B. NGUYEN, G. CHAUHAN, Y. HAO, A. MATHEWS, S. LI: *PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel*, 2023, arXiv: [2304.11277](https://arxiv.org/abs/2304.11277) [cs.DC], URL: <https://arxiv.org/abs/2304.11277>.
- [40] Y. ZHENG, R. ZHANG, J. ZHANG, Y. YE, Z. LUO: *LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models*, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), ed. by Y. CAO, Y. FENG, D. XIONG, Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 400–410, DOI: [10.18653/v1/2024.acl-demos.38](https://doi.org/10.18653/v1/2024.acl-demos.38), URL: <https://aclanthology.org/2024.acl-demos.38/>.