FMF-AI

FORMAL METHODS AND FOUNDATIONS
OF ARTIFICIAL INTELLIGENCE

Proceedings of the International Conference on Formal Methods and Foundations of Artificial Intelligence

Eszterházy Károly Catholic University,
Eger, Hungary, June 5–7, 2025

ESZTERHÁZY KÁROLY CATHOLIC UNIVERSITY

Proceedings of the International Conference on Formal Methods and Foundations of Artificial Intelligence

Eszterházy Károly Catholic University,
Eger, Hungary, June 5–7, 2025

Conference Chairman: Csaba Biró
Co-chairman: Olivér Hornyák
Secretary: Ferenc Koczka, Tibor Tajti
Chairman of the Organising Committee: Gábor Kusper
Head of the Program Committee: Gergely Kovásznai

Conference Chairman

- Csaba Biró (EKCU)

Co-chairman

- Olivér Hornyák (University of Miskolc)

Secretary

- Ferenc Koczka (EKCU)
- Tibor Tajti (EKCU)

Chairman of the Organising Committee

- Gábor Kusper (EKCU)

Members of the Organising Committee

- Ádám Kovács (EKCU)
- Ágnes Nagyné Bertalan (EKCU)
- Ede Troll (EKCU)
- Erzsébet Tóth (EKCU)
- Péter Takács (EKCU)
- Péter Szigetváry (EKCU)
- Roland Király (EKCU)
- Tamás Balla (EKCU)

Head of the Program Committee

- Gergely Kovásznai (EKCU)

Program Committee

- András Horváth (PPKE)
- András Kovács (HUN-REN – SZTAKI)
- András Vörös (BME)
- Anikó Apró (EKCU)
- Armin Biere (University of Freiburg)
- Balázs Harangi (University of Debrecen)
- Csaba Szabó (TUKE)
- Dongzhe Ren (Chongqing Aerospace Polytechnic)
- Erika Ábrahám (RWTH – Aachen)
- György Vaszil (University of Debrecen)
- Isabela Dramnesc (West University of Timisoara)
- Ismail Bogrekci (ADU)
- János Botzheim (ELTE)
- Károly Nehéz (University of Miskolc)
- László Kovács (University of Miskolc)
- Lehel Csató (Babes Bolyai University)
- Martina Seidl (JKU)
- Máté Tejfel (ELTE)
- Miklós Hoffmann (EKCU)
- Nikolaj Popov (JKU – RISC)
- Norbert Pataki (ELTE)
- Olivér Hornyák (University of Miskolc)
- Pál Varga (BME)
- Péter Sinčák (TUKE)
- Róbert Lovas (HUN-REN – SZTAKI)
- Sándor Jenei (EKCU)
- Sándor Király (EKCU)
- Sorin Stratulat (Univ. of Lorraine)
- Tamás Balla (EKCU/ELTE)
- Tibor Csendes (University of Szeged)
- Viktória Zsók (ELTE)
- Wolfgang Schreiner (JKU – RISC)
- Yang Győző (HUN-REN Hungarian Research Centre For Linguistics)
- Yungiao Yang (Chongqing Aerospace Polytechnic)
- Zoltán Dhulfikar (ELTE)
- Zoltán Istenes (ELTE)

https://uni-eszterhazy.hu/fmf

Our conference proceedings were created with the support of the MECENATÚRA project MEC_SZ_149312 titled “Formal Methods and Foundations of Artificial Intelligence” coordinated by Eszterházy Károly Catholic University.

PROGRAM FINANCED FROM THE NRDI FUND

AZ NKFI ALAPBÓL MEGVALÓSULÓ PROGRAM

EGER 1774 ESZTERHÁZY KÁROLY CATHOLIC UNIVERSITY

Mecenatúra

ISBN 978-963-496-303-5 (PDF)

Responsible for publication
Rector of Eszterházy Károly Catholic University
Published by Líceum Kiadó
Publishing director: Andor Nagy
Technical editor: Tibor Tómács
Cover design: Márton Dankó
Published: October 2025

Contents

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025

Philosophy and vision of the FMF-AI Conference series – A foreword

Csaba Biró[a], **Olivér Hornyák**[b], **Gergely Kovásznai**[a], **Gábor Kusper**[a]

[a]Eszterházy Károly Catholic University
Institute of Mathematics and Informatics, Eger, Hungary
{biro.csaba,kovasznai.gergely,kusper.gabor}@uni-eszterhazy.hu

[b]University of Miskolc, Institute of Information Science
oliver.hornyak@uni-miskolc.hu

Four years ago, we planted the first seed of what would later become the FMF-AI Conference. At that time, it was only a small, close-knit workshop – a gathering of researchers who shared a passion for exploring the formal and foundational aspects of Artificial Intelligence. Over the years, this seedling grew steadily: we held the workshop annually, nurturing its roots and community.

In 2025, thanks to the Mecenatúra Grant of NKFIH, Grant ID: MEC_SZ_149312, Title: Formal Methods and Foundations of Artificial Intelligence, this small plant finally blossomed into a full-grown tree: we brought the FMF-AI initiative into the light and opened our doors wide to hold the First International Conference on Formal Methods and Foundations of Artificial Intelligence (FMF-AI) at Eszterházy Károly Catholic University.

We would like to express our deep gratitude to the Mecenatúra Grant and the NKFIH for making this possible. We hope that this young tree – rooted in collaboration, nurtured by curiosity, and strengthened by shared knowledge – will continue to grow, withstand the storms ahead, and flourish into a strong, enduring tree in the years to come.

The conference in numbers and facts

The conference focused on the rapidly advancing field of Artificial Intelligence (AI), covering a broad spectrum of topics – from mathematical foundations and large language models to educational applications of AI. The event was enriched by distinguished invited speakers from both Hungary (e.g., from HUN-REN SZTAKI)

and abroad (e.g., from the West University of Timișoara).

Artificial Intelligence has become one of the most transformative forces of our time, reshaping how we work, learn, communicate, and make decisions. Its importance lies not only in its technological potential, but also in its capacity to address some of the most pressing challenges faced by humanity – from improving healthcare and sustainability to enhancing education and industrial efficiency. However, as AI systems become increasingly autonomous and pervasive, ensuring their transparency, reliability, and ethical grounding has become paramount. Conferences such as FMF-AI play a crucial role in fostering dialogue between theoreticians and practitioners, helping to bridge the gap between formal methods and real-world applications, and promoting the responsible and explainable use of Artificial Intelligence.

FMF-AI brought together 84 participants from 11 countries, featuring 6 invited talks, 1 roundtable discussion, 2 tutorials, 1 poster session, and 65 scientific presentations. One of the most memorable highlights was the roundtable discussion on the present and future of AI, addressing its challenges and opportunities. This session was broadcast online and remains available on the conference website.

As organizers, we were especially delighted to welcome many young researchers among the participants. To recognize their contribution, we established the Youngest Researcher of the Day Award, which was presented on each of the three conference days. It was equally inspiring to witness the enthusiastic exchanges between experienced and early-career researchers during the coffee breaks.

The conference received 72 submissions in total. After a rigorous two-round peer review process, 32 submissions were rejected, resulting in an overall rejection rate of 44.44 %. The 20 accepted papers included in this proceedings volume reflect both the quality and diversity of the conference contributions.

These papers span a wide range of topics – from the practical applications of AI, such as machine learning in robotics, image processing, and education, to the theoretical foundations of artificial intelligence, including formal models, explainability, and AI ethics. Together, they capture the interdisciplinary essence of FMF-AI, bridging theory and practice in the study of intelligent systems.

We thank the authors for their contributions and the invited speakers for enriching the conference with their perspectives. This proceedings volume reflects the collaborative spirit that defines FMF-AI – a community where formal reasoning and artificial intelligence converge to advance knowledge and serve society.

We extend our sincere thanks to the members of the Program Committee for their essential role in shaping this conference. Their diligent review of submissions and expert judgment were critical in assembling a high-quality and diverse program. We are grateful for their significant contribution of time and expertise.

In addition to this proceedings, a special journal issue, *Annales Mathematicae et Informaticae*, see: https://ami.uni-eszterhazy.hu/, featuring another 20 peer-reviewed papers will further disseminate the most outstanding results presented at the conference.

We hope that readers will find in these pages both inspiration and guidance for

their own research, and that this collection of papers will contribute meaningfully to the growing body of scientific work on the foundations and applications of Artificial Intelligence.

Looking ahead, our goal is to make FMF-AI an annual international conference, hosted each year at a different academic venue to strengthen international collaboration and visibility. While we will not reveal the exact location yet, we hope to meet again next year in France. **Stay tuned!**

Sincerely,

Csaba Biró, Conference Chair
Olivér Hornyák, Conference Co-Chair
Gergely Kovásznai, Chair of the Program Committee
Gábor Kusper, Chair of the Organizing Committee

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 4–13
DOI: 10.17048/fmfai.2025.4

Efficiency analysis comparison of the Particle Swarm Optimization and Tabu Search in Flow Shop Scheduling Problem*

Anita Agárdi

Institute of Informatics, University of Miskolc
anita.agardi@uni-miskolc.hu

Abstract. The paper investigates the efficiency of two metaheuristic algorithms, the Particle Swarm Optimization and the Tabu Search, on the Flow Shop Scheduling Problem. Particle Swarm Optimization is a population algorithm. The algorithm maintains a population of solutions. It improves the population during the iteration. Tabu Search improves a single possible solution. The paper presents the efficiency of the algorithms on a benchmark dataset and compares it with results published by other researchers.

1. Introduction

Industry 4.0 [4] is also known as the fourth industrial revolution. It has fundamentally changed manufacturing and industrial processes. It integrates digitalization, automation and artificial intelligence. This new approach enables production systems to become smarter. Machines and equipment no longer perform only traditional, pre-programmed tasks. It can process data in real time and perform analyses. It also can make autonomous decisions to increase efficiency. The principles of Industry 4.0 is the closer connection of information technology and industrial processes. Every element of the production chain can be continuously monitored, analysed and optimised. Internet of Things (IoT), cloud computing, robotics, augmented reality and data analytics are important in this process.

Production scheduling [7] aims to efficiently allocate and utilize available resources, such as machinery, labor, and materials. Production scheduling involves

*Supported by the University Research Scholarship Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

scheduling production orders, setting priorities etc. This process also takes into account production capacities, machine maintenance needs and order deadlines. The aim of production scheduling is the downtime minimization, reducing unnecessary inventory accumulation.

Metaheuristic algorithms [1] are optimization methods. The aim is to find near-optimal solutions to complex problems. These algorithms do not guarantee finding the globally optimal solution. The algorithms try to navigate the solution space efficiently. They usually combine random search elements with structured, heuristic methods. They are able to avoid local optima and find solution near the global optimum. For example, Genetic Algorithm [13], Simulated Annealing [5], Tabu Search [2], Particle Swarm Optimization [14], etc.

2. Materials and methods

2.1. Flow Shop Scheduling

Flow Shop Scheduling (FSS) [6] is a production scheduling problem. During the problem, products or workpieces must pass through different work processes or machines. The objective function of the problem is minimizing production time, downtime, and/or delays. All workpieces pass through the same machines and operations. The processing times assigned to each machine can be different. Establishing the correct sequence and schedule can be very difficult. Formally, a Flow Shop Scheduling problem can be defined as follows:

Consider a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ and a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each job J_i consists of a sequence of m operations, one for each machine, with processing time $p_{i,j}$ on machine M_j. The objective is to find a permutation π of the jobs that minimizes a given criterion, such as the makespan, total completion time, or total tardiness.

Mathematically, the classical Flow Shop Scheduling problem with makespan minimization can be written in a following way:

$$C_{\max} = \max_{i=1,\ldots,n} \{C_{i,m}\},$$

where $C_{i,j}$ is the completion time of job J_i on machine M_j, calculated recursively as:

$$\begin{cases} C_{1,1} = p_{1,1}, & \\ C_{i,1} = C_{i-1,1} + p_{i,1}, & i = 2, \ldots, n, \\ C_{1,j} = C_{1,j-1} + p_{1,j}, & j = 2, \ldots, m, \\ C_{i,j} = \max(C_{i-1,j}, C_{i,j-1}) + p_{i,j}, & i = 2, \ldots, n,\ j = 2, \ldots, m. \end{cases}$$

Typical examples of Flow Shop Scheduling applications are:

- Automotive assembly: Parts are assembled in a specific order.
- Electronics production: Testing, assembling, and quality control of components.

- Metalworking: Cutting, welding, and grinding of metals.

2.2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) [3] is a population-based optimization algorithm. The algorithm is inspired by the movement of particles. It was developed by James Kennedy and Russell Eberhart in 1995. Particles represent potential solutions that move through the search space based on their own experience and that of the swarm. Steps of the algorithm:

1. **Particle creation:** Each particle represents a possible solution. The elements of the population can be randomly generated or can be the results of a construction algorithm.

2. **Velocity and position update:** Particles get new velocity and position in each iteration. Velocity is updated based on the following formula:

$$\vec{v}_i(t+1) = w \cdot \vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{pbest}_i - \vec{x}_i(t)) + c_2 \cdot r_2 \cdot (\vec{gbest} - \vec{x}_i(t))$$

   where:

   $\boldsymbol{\vec{v}_i(t)}$**:** velocity of particle i in iteration t

   $\boldsymbol{\vec{x}_i(t)}$**:** position of particle i in iteration t

   $\boldsymbol{w}$**:** inertia weight

   $\boldsymbol{c_1}$**,** $\boldsymbol{c_2}$**:** learning factors that influence attraction towards $\vec{pbest}_i$ and $\vec{gbest}$

   $\boldsymbol{\vec{pbest}_i}$**:** best location of particle i

   $\boldsymbol{\vec{gbest}}$**:** global best solution

   $\boldsymbol{r_1}$ **and** $\boldsymbol{r_2}$**:** are random numbers between $[0, 1]$. They are independently regenerated for each particle in every iteration. Their role is to provide a random weighting for the particle's movement. r_1 scales the "cognitive" component, pulling the particle towards its own best position ($\vec{pbest}_i$). r_2 scales the "social" component, pulling the particle towards the global best position ($\vec{gbest}$).

   The position of particles is updated with the following formula:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1)$$

3. **Termination condition:** The algorithm stops if the termination condition is met. The termination condition can be a certain number of iterations, convergence, or a fixed runtime.

Particle Swarm Optimization was originally developed for continuous tasks, but the Flow Shop Scheduling task is a discrete problem, so the algorithm needs to be discretized. The discretization was based on the article [10], which solves the Traveling Salesman Problem.

Algorithm 1: Discrete Particle Swarm Optimization for Flow Shop Scheduling.

Input: Flow Shop Scheduling problem instance
Output: Best found solution

1. 1. Create the initial particles. The particles' positions $\vec{x}_i(t)$ represent permutations in the Flow Shop Scheduling problem. In this case, they are generated randomly.
2. 2. Initialize the particles' velocities $\vec{v}_i(t)$. This is a Basic Swap Sequence value [10], since the problem is discrete.
3. **while** *termination criteria is not met* **do**
4. 2.a Compute the global best particle.
5. **foreach** *particle* **do**
6. 3. Compute the particle velocity using the following formula:

$$\vec{v}_i(t+1) = \vec{v}_i(t) \oplus c_1 r_1(\vec{pbest}_i - \vec{x}_i(t)) \oplus c_2 r_2(\vec{gbest} - \vec{x}_i(t))$$

7. Here, the Basic Swap Sequence [10] represents the sequence of swaps between:
8. – the best position of the particle $\vec{pbest}_i$ and its current position (both are permutations) ie. $\vec{pbest}_i - \vec{x}_i(t)$, and
9. – the global best $\vec{gbest}$ and the particle's current position ie. $\vec{gbest} - \vec{x}_i(t)$.
10. The operator $\oplus$ denotes performing swaps between permutations (adding Basic Swap Sequence values).
11. 4. Determine the current particle's new position using the following formula:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1)$$

This means performing the swaps defined by the Basic Swap Sequence on the permutation.
12. 5. Determine the fitness value of $\vec{x}_i(t+1)$;

2.3. Tabu Search

Tabu Search (TS) [12] is a local search procedure. The algorithm maintains a tabu list to move from a local optimum to a global optimum.

The steps of the algorithm are the followings:

- **Initial solution:** Taking an initial solution (it is either randomly generated or constructed using some construction technique). This solution will be the current solution. In the case of Flow Shop Scheduling Problem each solution means a randomly generated permutation.
- **Neighborhood search:** Searching for the neighbors of the current solution

using a neighborhood operator. In the case of Flow Shop Scheduling Problem, the swap operator is used for this.

- **Tabu list:** The tabu list stores those solutions that cannot be re-selected (already visited solutions).
- **Recording best solution:** The algorithm continuously monitors the best solution found so far.
- **Deleting individual elements of the tabu list:** If the tabu list is full, the older solutions are deleted.
- **Termination condition:** The termination condition can be a certain number of iterations, convergence, or a predefined runtime.

3. Test results

This section contains the test results. First, the results of Particle Swarm Optimization, then the Tabu Search test runs are presented.

Separate tables are provided for both PSO and TS algorithms, showing the maximum, average, and minimum fitness values of the runs for each Taillard data set (Table 1, Table 3). The article also examines the fitness values of the proposed algorithms in comparison with the already published data. Table 2 and Table 4 contain the fitness values of the PSO or TS and and how much better the TS or PSO algorithm is than other algorithms published in the literature. If the percentage is positive, then the PSO or TS algorithm is better than the given comparison algorithm. The comparisons were created with the published results of the following algorithms [8, 11]:

- HMM-PFA – Hidden Markov Model based Particle Filter Algorithm
- HGA – Hybrid Genetic Algorithm
- IIGA – Improved Invasive Weed Optimization Algorithm
- DSOMA – Differential Search Optimization Method Algorithm
- HGSA – Hybrid Gravitational Search Algorithm

The Taillard [9] dataset was used during the test runs. The key features of the Taillard dataset is the following:

- Variety of problem sizes: The dataset includes examples for different numbers of machines (m) and workpieces (n) (e.g., 20×5, 50×10, 100×20), allowing testing of small and large scale problems.
- Independent, randomly generated processing times: The processing times are chosen so that the problems are not biased or trivially solvable.

- Wide acceptance: The benchmark is a quasi-standard in FSSP research and serves as a reference for testing almost all modern metaheuristic and exactness algorithms.
- The objective function of the problems is the makespan minimization

3.1. Particle Swarm Optimization test results

This section presents the results of Particle Swarm Optimization. First, the maximum, average, and minimum of the test runs are presented. Then, the section examines the test results in comparison to the benchmark algorithms.

Table 1. Fitness values of Particle Swarm Optimization: maximum, average and minimum values.

Instance	**PSO**		
	Max	Avg	Min
Ta001	1313	1301.8	1297
Ta002	1373	1368.4	1366
Ta003	1161	1153.2	1145
Ta004	1391	1380.6	1372
Ta005	1288	1284.6	1277
Ta006	1258	1248	1238
Ta007	1273	1262.4	1252
Ta008	1297	1281.6	1271
Ta009	1298	1288.2	1277
Ta010	1177	1170.6	1161
Ta011	1725	1713	1708
Ta012	1792	1785.4	1778
Ta013	1627	1616.2	1599
Ta014	1509	1493.8	1469
Ta015	1562	1552.6	1546
Ta016	1530	1509.6	1493
Ta017	1599	1586.8	1571
Ta018	1679	1667.6	1631
Ta019	1701	1693	1686
Ta020	1728	1711	1692

The Table 1 shows the performance of the Particle Swarm Optimization (PSO) algorithm on the Taillard dataset. The average fitness values of the solutions is close to the best value. This means, that PSO gives relatively stable results. However, differences can be observed between individual instances. For example, in the case of Ta018, where the difference between the Max and Min values is large (1679 and 1631 fitness values). In the case of Ta011, the difference is minimal. The algorithm achieved the lowest values for problems Ta003 (minimum fitness value: 1145) and

Ta010 (minimum fitness value: 1161). The worst result was on problem Ta012 (maximum fitness value: 1792).

Table 2. Comparison of test results obtained by Particle Swarm Optimization and competing algorithms.

Instance	PSO	HMM-PFA %	HGA %	IIGA %	DSOMA %	HGSA %
Ta001	1297	14.57	11.72	14.57	5.94	2.08
Ta002	1366	11.86	6.88	11.86	3.07	5.56
Ta003	1145	27.51	21.05	27.51	11.79	-4.10
Ta004	1372	15.74	10.86	15.74	5.54	7.07
Ta005	1277	13.47	9.87	13.47	5.01	1.10
Ta006	1238	19.63	15.51	19.63	10.10	12.36
Ta007	1252	18.45	16.69	18.45	10.30	3.75
Ta008	1271	16.60	12.75	16.60	8.50	1.65
Ta009	1277	15.04	9.48	15.04	7.52	2.27
Ta010	1161	18.60	14.04	18.60	10.51	6.20
Ta011	1708	19.67	14.46	17.74	-0.59	0.29
Ta012	1778	21.82	19.40	21.82	3.09	-3.37
Ta013	1599	21.33	19.57	21.33	4.82	-2.75
Ta014	1469	23.28	21.31	23.28	5.24	3.20
Ta015	1546	25.03	25.03	25.03	4.59	1.75
Ta016	1493	26.72	22.37	26.72	6.50	-2.41
Ta017	1571	24.95	23.74	24.95	3.25	3.25
Ta018	1631	26.12	22.99	26.12	6.13	7.23
Ta019	1686	17.02	13.17	17.02	3.62	-3.68
Ta020	1692	21.22	18.26	21.22	5.32	1.77

Table 2 compares the maximum of the test values of the Particle Swarm Optimization algorithm and the results of the algorithms published by the researchers. In some cases, the HGSA algorithm and in one case the DSOMA algorithm gave better results than the PSO algorithm. In the majority of cases (in the range of about 10-20 %) PSO gave better results than the other algorithms. In some cases (e.g. Ta001, Ta006, Ta018) HGSA gave better or similar results than PSO. In one case (Ta003) DSOMA achieved better results (-4.10 %) than PSO.

3.2. Tabu Search test results

In the following, the test results of the Tabu Search algorithm and their comparison with the results of algorithms published by researchers are presented.

The results of Tabu Search are presented in Table 3. The results show that the algorithm typically provides stable and reliable performance. In most cases, the maximum and average values are relatively close to each other. The method usually gives consistently good results. For example, for problem Ta006, the best fitness value is 1296, the average value is 1272.2, and the worst is 1233.

Table 4 compares the Tabu Search results with the published results. Only five cases was the proposed algorithm worse than the published results. In many cases,

Table 3. Fitness values of Tabu Search algorithm: maximum, average and minimum values.

Instance	**TS**		
	Max	Avg	Min
Ta001	1377	1356.6	1323
Ta002	1447	1405.6	1383
Ta003	1198	1166.2	1132
Ta004	1444	1405	1359
Ta005	1416	1333.2	1279
Ta006	1296	1272.2	1233
Ta007	1294	1275	1259
Ta008	1297	1269.2	1226
Ta009	1376	1321.6	1265
Ta010	1180	1162.4	1137
Ta011	1760	1723.4	1681
Ta012	1852	1784.6	1737
Ta013	1656	1613.8	1591
Ta014	1609	1518.6	1425
Ta015	1593	1553.6	1495
Ta016	1515	1487.8	1467
Ta017	1649	1611.2	1576
Ta018	1722	1679.8	1641
Ta019	1745	1701.8	1672
Ta020	1751	1720.2	1657

the TS algorithm gave more than 20% better results. The IIGA and HMM-PFA algorithms were similar or worse than TS. The DSOMA and HGSA algorithms gave significantly worse results. In most cases, TS gave results that were more than 20% better than the other algorithms, for example, for Ta003, IIGA is 28.98%. TS only performed worse in five cases: for Ta011 and Ta012, in the case of HGSA and DSOMA, and for Ta019 and Ta016, in the case of HGSA and DSOMA.

4. Conclusions and future research directions

The paper investigated the effectiveness of Particle Swarm Optimization and Tabu Search on the Flow Shop Scheduling Problem. The Taillard benchmark dataset was used for the problem. It can be said that TS was the most efficient algorithm. Most of the compared algorithms, such as HMM-PFA, HGA, IIGA, performed worse than the proposed algorithms. Particle Swarm Optimization (PSO) and Tabu Search (TS) algorithms give several future research directions, for example the hybridization of PSO and TS. Further research can also focus on combining them with other metaheuristic algorithms, such as Genetic Algorithm or Simulated

Table 4. Comparison of test results obtained by Tabu Search and competing algorithms.

Instance	TS	HMM-PFA %	HGA %	IIGA %	DSOMA %	HGSA %
Ta001	1323	12.32	9.52	12.32	3.85	0.08
Ta002	1383	10.48	5.57	10.48	1.81	4.27
Ta003	1132	28.98	22.44	28.98	13.07	-3.00
Ta004	1359	16.85	11.92	16.85	6.55	8.09
Ta005	1279	13.29	9.70	13.29	4.85	0.94
Ta006	1233	20.11	15.98	20.11	10.54	12.81
Ta007	1259	17.79	16.04	17.79	9.69	3.18
Ta008	1226	20.88	16.88	20.88	12.48	5.38
Ta009	1265	16.13	10.51	16.13	8.54	3.24
Ta010	1137	21.11	16.45	21.11	12.84	8.44
Ta011	1681	21.59	16.30	19.63	1.01	1.90
Ta012	1737	24.70	22.22	24.70	5.53	-1.09
Ta013	1591	21.94	20.18	21.94	5.34	-2.26
Ta014	1425	27.09	25.05	27.09	8.49	6.39
Ta015	1495	29.30	29.30	29.30	8.16	5.22
Ta016	1467	28.97	24.54	28.97	8.38	-0.68
Ta017	1576	24.56	23.35	24.56	2.92	2.92
Ta018	1641	25.35	22.24	25.35	5.48	6.58
Ta019	1672	18.00	14.11	18.00	4.49	-2.87
Ta020	1657	23.78	20.76	23.78	7.54	3.92

Annealing. Optimization of PSO and TS parameters can also be another future research direction.

References

[1] P. Agrawal, H. Abutarboush, T. Ganesh, A. Mohamed: *Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)*, IEEE Access 9 (2021), pp. 26766–26791, doi: 10.1109/ACCESS.2021.3056407.

[2] O. Arik: *Population-based Tabu search with evolutionary strategies for permutation flow shop scheduling problems under effects of position-dependent learning and linear deterioration*, Soft Computing 25.2 (2021), pp. 1501–1518, doi: 10.1007/s00500-020-05234-7.

[3] A. Gad: *Particle swarm optimization algorithm and its applications: a systematic review*, Archives of Computational Methods in Engineering 29.5 (2022), pp. 2531–2561, doi: 10.1007/s11831-021-09694-4.

[4] H. Lasi, P. Fettke, H. Kemper, T. Feld, M. Hoffmann: *Industry 4.0*, Business & Information Systems Engineering 6 (2014), pp. 239–242, doi: 10.1007/s12599-014-0334-4.

[5] Y. Li, C. Wang, L. Gao, Y. Song, X. Li: *An improved simulated annealing algorithm based on residual network for permutation flow shop scheduling*, Complex & Intelligent Systems 7 (2021), pp. 1173–1183, doi: 10.1007/s40747-020-00205-9.

[6] J. Neufeld, S. Schulz, U. Buscher: *A systematic review of multi-objective hybrid flow shop scheduling*, European Journal of Operational Research 309.1 (2023), pp. 1–23, doi: 10.1016/j.ejor.2022.08.009.

[7] M. Parente, G. Figueira, P. Amorim, A. Marques: *Production scheduling in the context of Industry 4.0: review and trends*, International Journal of Production Research 58.17 (2020), pp. 5401–5431, doi: 10.1080/00207543.2020.1718794.

[8] C. Qu, Y. Fu, Z. Yi, J. Tan: *Solutions to no-wait flow shop scheduling problem using the flower pollination algorithm based on the hormone modulation mechanism*, Complexity (2018), doi: 10.1155/2018/1973604.

[9] E. Taillard: *Benchmarks for basic scheduling problems*, European Journal of Operational Research 64.2 (1993), pp. 278–285, doi: 10.1016/0377-2217(93)90182-M.

[10] K.-P. Wang, L. Huang, C.-G. Zhou, W. Pang: *Particle swarm optimization for traveling salesman problem*, in: Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE cat. no. 03ex693), vol. 3, IEEE, 2003, pp. 1583–1585.

[11] H. Wei, S. Li, H. Jiang, J. Hu: *Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion*, Applied Sciences 8.12 (2018), p. 2621, doi: 10.3390/app8122621.

[12] J. Xie, X. Li, L. Gao, L. Gui: *A hybrid genetic tabu search algorithm for distributed flexible job shop scheduling problems*, Journal of Manufacturing Systems 71 (2023), pp. 82–94, doi: 10.1016/j.jmsy.2023.09.002.

[13] W. Xu, L. He, G. Zhu: *Many-objective flow shop scheduling optimisation with genetic algorithm based on fuzzy sets*, International Journal of Production Research 59.3 (2021), pp. 702–726, doi: 10.1080/00207543.2019.1705418.

[14] W. Zhang, W. Hou, C. Li, W. Yang, M. Gen: *Multidirection update-based multiobjective particle swarm optimization for mixed no-idle flow-shop scheduling problem*, Complex System Modeling and Simulation 1.3 (2021), pp. 176–197, doi: 10.23919/CSMS.2021.0017.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 14–22
DOI: 10.17048/fmfai.2025.14

Applications of machine learning in underwater bioacoustics*

Attila Aradi[a†], Péter Takács[b], Attila Károly Varga[c]

[a]ION-Technik Kft.
aradi.attila@ion-technik.hu

[b]HUN-REN Balaton Limnological Research Institute
peter.takacs@blki.hu

[c]University of Miskolc
attila.varga@uni-miskolc.hu

Abstract. Underwater bioacoustics, the study of sound in aquatic biological systems, is increasingly enhanced by machine learning (ML) technologies. This paper explores recent developments in applying ML to underwater bioacoustics, focusing on marine and freshwater species identification, environmental monitoring, and noise reduction. We examine key methodologies, present performance analysis from various applications, and address the challenges unique to the underwater domain. Additionally, we propose future directions for research including multimodal approaches and real-time processing systems.

Keywords: underwater bioacoustics, machine learning, species identification, environmental monitoring, convolutional neural networks

AMS Subject Classification: 68T07, 92D40, 94A12

1. Introduction

Underwater bioacoustics investigates the production, propagation, and perception of sound by aquatic organisms. This field is essential for understanding marine and freshwater life behavior, communication, and ecological dynamics. Sound plays a

*The first author was supported by the KDP Kooperatív Doktori Program, Kultúrális és Innovációs Minisztérium, Nemzeti Kutatási, Fejlesztési és Innovációs Alap, and ION-Technik Kft.
†Corresponding author.

crucial role in aquatic environments where visual information is often limited by depth, turbidity, or lighting conditions.

However, collecting and analyzing underwater acoustic data presents significant challenges due to signal distortion, background noise, and the diversity of sound sources. Traditional manual analysis methods are time-consuming and often impractical for large-scale monitoring efforts. The acoustic environment underwater is complex, with sounds from biological sources, geological activities, weather conditions, and increasing anthropogenic noise pollution.

Machine learning offers promising tools to overcome these issues by automating detection, classification, and interpretation of bioacoustic signals [7]. The application of ML in underwater bioacoustics has grown significantly in recent years, driven by advances in deep learning architectures and the availability of larger acoustic datasets. These developments enable researchers to process acoustic data with improved accuracy and efficiency, opening new possibilities for monitoring and conservation applications.

2. Machine learning methodologies in underwater bioacoustics

2.1. Species identification and classification

Supervised ML algorithms, especially convolutional neural networks (CNNs), have been widely used for classifying species-specific vocalizations. These models are trained on spectrograms derived from hydrophone recordings, leveraging pattern recognition capabilities to identify acoustic signatures. The transformation of temporal acoustic signals into spectrograms creates a visual representation that captures both frequency content and temporal dynamics.

Recent implementations have achieved notable classification accuracy. Researchers have successfully detected dolphin clicks, whale songs, fish choruses, and freshwater species such as frogs and riverine fish using deep learning approaches [2, 10]. Transfer learning approaches, where models pre-trained on general datasets are adapted for bioacoustic spectrograms, have proven effective when labeled acoustic data is limited [3].

Different species present varying challenges for detection algorithms. Dolphin echolocation clicks, characterized by their brief duration and high frequency content, require specialized temporal processing. Whale songs, with their complex hierarchical structure, benefit from models capable of understanding long-term dependencies. Fish choruses during spawning seasons create complex acoustic landscapes where multiple species vocalize simultaneously, necessitating advanced separation techniques.

2.2. Environmental monitoring and ecosystem assessment

ML is used to monitor aquatic environments by detecting biotic and anthropogenic sounds. In both marine and freshwater habitats, anomaly detection models can identify changes in acoustic environments due to pollution, vessel traffic, or climate effects. Long Short-Term Memory (LSTM) networks and transformer models have proven effective in identifying temporal patterns across different time scales [6].

Soundscape ecology applications use ML to examine the acoustic environment as an indicator of ecological health. Automated species detection enables calculation of acoustic diversity indices, providing assessments of ecosystem health that complement traditional surveys. These approaches can track changes over time with high temporal resolution.

The detection of anthropogenic impacts represents a critical application area. Vessel noise pollution, construction activities, and other human activities create distinct acoustic signatures that ML models can detect and quantify [5]. This capability enables assessment of human impacts on marine ecosystems and supports management decisions.

2.3. Noise reduction and signal enhancement

Underwater recordings are often degraded by complex noise sources such as boat engines, wave motion, or flow-induced turbulence in rivers and lakes. ML models, including denoising autoencoders and non-negative matrix factorization (NMF), can isolate biological signals from noise. These techniques improve the reliability of ecological interpretations [9].

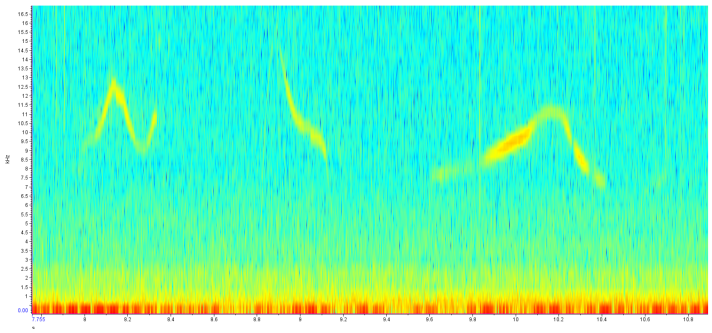


Figure 1. Example spectrogram of underwater acoustic data with visible animal vocalizations in the 10–13 kHz range. Such features are often targeted by machine learning models for detection and classification tasks.

Denoising autoencoders learn to map corrupted signals back to their original form by training on pairs of noisy and clean acoustic data. These models can effectively remove various types of noise while preserving essential characteristics

of biological signals. The encoder-decoder architecture enables learning of complex mappings that traditional filters cannot achieve.

Recent advances include generative adversarial networks (GANs) for signal enhancement, where competing networks learn to generate clean signals and distinguish between real and generated outputs. This approach shows promise for removing non-stationary noise sources that vary over time. Traditional spectral subtraction and Wiener filtering methods have also been enhanced through ML-based parameter optimization [1, 9].

3. Performance analysis and applications

3.1. Marine environment applications

Studies in marine environments have demonstrated the effectiveness of ML approaches for large-scale monitoring with quantitative improvements over traditional methods. Shiu et al. [10] reported a multi-species cetacean detection system achieving mean average precision (mAP) of 0.87 across 15 species, with individual species ranging from 0.72 (beaked whales) to 0.94 (humpback whales). Their CNN-based approach processed 187,000 hours of recordings from the Pacific Ocean, demonstrating scalability for large-scale monitoring efforts.

Bermant et al. [2] developed a deep learning system for beluga whale detection achieving 97.5% precision and 94.8% recall on a test set of 5,840 calls. The system maintained 91.2% accuracy when deployed in different geographic locations, demonstrating cross-region generalization capabilities. Environmental factors significantly influenced performance, with detection accuracy dropping to 78.4% in high noise conditions (SNR $< 10\,\mathrm{dB}$).

For dolphin echolocation clicks, recent implementations achieved F1 scores of 0.89–0.93 using ResNet architectures [3]. Detection performance varied with click train characteristics: isolated clicks (precision: 0.91, recall: 0.88) versus overlapping click trains (precision: 0.84, recall: 0.79). Processing speeds reached 450× real-time on GPU hardware, enabling efficient analysis of long-term recordings. Multi-species detection systems have been developed capable of identifying multiple cetacean species from continuous recordings. These systems typically employ hierarchical classification, first detecting the presence of marine mammal vocalizations, then applying species-specific models.

Performance varies across species, with larger whales generally showing higher detection rates due to their distinctive vocalizations. Smaller dolphins and porpoises present greater challenges due to overlapping frequency ranges and variable acoustic signatures. Environmental factors such as ambient noise levels and propagation conditions significantly influence detection performance [2, 5].

3.2. Freshwater monitoring systems

Freshwater environments present unique challenges due to species diversity and variable acoustic conditions. ML-based fish monitoring systems have been developed for species detection during spawning seasons when acoustic activity peaks. Custom architectures account for the specific propagation characteristics and noise sources in shallow water environments.

Success rates vary among species, with those producing distinctive sounds achieving higher detection accuracy. Environmental factors including wind-generated noise, thermal stratification, and human activities affect system performance. Adaptive algorithms that adjust detection thresholds based on ambient conditions have improved robustness [1, 4].

3.3. Real-time processing implementations

Real-time monitoring systems demonstrate practical deployment capabilities with quantifiable performance trade-offs. Edge computing implementations using optimized neural networks achieved 82–89% of full model accuracy while reducing computational requirements by 75%. Briggs et al. [4] deployed autonomous buoys processing 24 kHz audio continuously for 6 months, detecting target species with 86.3% accuracy using models compressed to 2.4 MB.

Lightweight architectures such as MobileNet variants maintained detection F1 scores above 0.80 while operating within 5 W power budgets. Processing latency ranged from 50–200 ms per 1-second audio segment on embedded platforms (NVIDIA Jetson series), enabling near real-time alerts for conservation applications. Battery-powered systems achieved 3–6 month deployment durations with solar charging, processing 8–16 hours daily [3]. Real-time monitoring systems have been deployed using edge computing platforms to process acoustic data continuously. These systems balance computational constraints with monitoring requirements, achieving acceptable performance for ecosystem-level assessment while operating within power and processing limitations.

Lightweight neural networks optimized for low-power consumption enable continuous operation on autonomous platforms. Despite computational constraints, these systems provide valuable insights into ecosystem dynamics and can detect significant changes in acoustic patterns [3, 4].

4. Challenges and limitations

4.1. Data-related challenges

The underwater environment introduces unique obstacles for ML applications. A major challenge is the scarcity of labeled datasets for marine and freshwater bioacoustics, which limits supervised training effectiveness. Data collection requires specialized equipment and often lengthy field campaigns, resulting in smaller datasets compared to other ML domains.

Data quality issues compound scarcity problems. Underwater recordings are affected by equipment limitations, environmental variability, and temporal constraints. Many species are only acoustically active during specific seasons or conditions, limiting representative training data availability.

Annotation quality presents additional challenges. Manual labeling requires expertise in both target species and acoustic analysis. Variability between annotators can be substantial, particularly for subtle vocalizations or overlapping calls from multiple species. Standardized annotation protocols and quality control measures are essential for reliable training data [6, 8].

4.2. Generalization and adaptation

Several strategies have been developed to address generalization challenges in underwater bioacoustics. Domain adaptation techniques using adversarial training improved cross-region performance by 15–22% for marine mammal detection tasks [10]. Unsupervised domain adaptation methods, requiring only unlabeled data from target environments, achieved 78–85% of supervised performance levels.

Transfer learning approaches demonstrate varying success rates depending on source-target similarity. Models pre-trained on terrestrial bird vocalizations and fine-tuned for marine mammals achieved 82% of purpose-trained model performance with 60% less training data. Within-domain transfer (e.g., between cetacean species) showed better results, reaching 91–95% of baseline performance [3].

Data augmentation strategies specifically designed for underwater acoustics include: Time-frequency masking: improved generalization by 8–12%. Noise injection using real environmental recordings: 10–15% improvement. Pitch shifting within species-specific ranges: 5–8% improvement. Simulated propagation effects: 12–18% improvement for depth-variant deployments. Models trained in specific regions or conditions often fail to generalize to new environments, limiting applicability across different ecosystems. This challenge is acute in underwater bioacoustics due to high variability in acoustic environments caused by bathymetry, substrate composition, and local noise sources.

Geographic variation in species vocalizations presents additional generalization challenges. Many species exhibit regional variations in acoustic signatures, requiring models to adapt to these differences. Transfer learning approaches show promise but require careful consideration of domain similarities and differences.

Seasonal and temporal variations further complicate generalization. Models trained on recordings from one season may perform poorly on data from different periods due to changes in species behavior, ambient noise, and acoustic propagation characteristics [8].

4.3. Technical and deployment constraints

Real-time processing requirements present computational challenges for many monitoring applications. Underwater platforms often have limited power and process-

ing resources, requiring efficient algorithms that operate within these constraints. Specialized hardware approaches are being explored to address these limitations.

The underwater environment creates unique deployment challenges. Equipment must withstand harsh conditions including pressure, corrosion, and biofouling. Communication limitations restrict data transmission capabilities, requiring on-board processing and compression techniques.

Maintenance and calibration difficulties affect long-term deployments. Unlike terrestrial systems, underwater platforms are difficult to access for routine maintenance, requiring robust designs and remote diagnostic capabilities.

5. Future directions

5.1. Methodological advances

Future research directions include developing more sophisticated architectures for underwater-specific challenges. Attention-based models show promise for capturing long-range dependencies in complex vocalizations. Self-supervised learning approaches may address data scarcity by learning representations without extensive manual labeling [3, 10].

Multimodal approaches that combine acoustic data with other sensor modalities offer potential for improved monitoring capabilities. Integration of acoustic recordings with environmental sensors and visual data could provide more comprehensive ecosystem insights [1, 5].

5.2. Technology integration

The development of edge computing solutions will enable more sophisticated real-time processing on autonomous platforms. Integration with distributed monitoring networks could create comprehensive systems that adapt to changing conditions and species distributions.

Cloud computing integration may enable advanced post-processing and analysis of data from multiple sources, identifying patterns and trends across larger spatial and temporal scales.

5.3. Conservation applications

Integration of ML outputs into real-time monitoring systems could enhance conservation efforts by providing immediate ecological insights. Early warning systems for environmental threats or species changes could enable rapid response measures.

Citizen science applications using simplified ML models could expand monitoring coverage while engaging public participation. Predictive modeling approaches combining species detection with environmental forecasting may enable proactive conservation measures.

6. Conclusion

Machine learning is transforming underwater bioacoustics by enabling automated analysis of marine and freshwater acoustic data. The field has progressed from manual analysis methods to sophisticated systems capable of species detection, ecosystem monitoring, and environmental assessment. As datasets grow and computational methods advance, ML will play an increasingly important role in aquatic ecology and conservation.

The integration of ML techniques with underwater bioacoustics has shown success across diverse applications, from species identification to ecosystem health monitoring. However, significant challenges remain, including data scarcity, generalization difficulties, and computational constraints for deployment.

Future developments will likely focus on multimodal systems, real-time processing capabilities, and conservation applications. As pressures on aquatic ecosystems continue to increase, these technological advances will become increasingly valuable for understanding, monitoring, and protecting aquatic environments.

Acknowledgements. The authors thank ION-technik Ltd. and ION Applied Science Ltd.

References

[1] K. J. Benoit-Bird, W. W. L. Au: *Extreme diel horizontal migrations by a tropical nearshore resident micronekton community*, Marine Ecology Progress Series 319 (2006), pp. 1–14, doi: 10.3354/meps319001.

[2] P. C. Bermant, M. M. Bronstein, R. J. Wood, S. Gero, D. Floreano: *Deep machine learning techniques for the detection and classification of sperm whale bioacoustics*, Scientific Reports 9.1 (2019), p. 12588, doi: 10.1038/s41598-019-48909-4.

[3] L. Bouffaut, N. Taburet, A. Menard, O. Dufour, C. Gervaise: *Deep-learning based detection and classification of blue whale vocalizations*, The Journal of the Acoustical Society of America 149.4 (2021), pp. 2605–2614, doi: 10.1121/10.0004495.

[4] F. Briggs, B. Lakshminarayanan, L. Neal, X. Z. Fern, R. Raich, S. J. K. Hadley, A. S. Hadley, M. G. Betts: *Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach*, The Journal of the Acoustical Society of America 131.6 (2012), pp. 4640–4650, doi: 10.1121/1.4707424.

[5] M. Castellote, C. W. Clark, M. O. Lammers: *Acoustic and behavioural changes by fin whales (Balaenoptera physalus) in response to shipping and airgun noise*, Biological Conservation 147.1 (2012), pp. 115–122, doi: 10.1016/j.biocon.2011.12.021.

[6] C. Gervaise, Y. Simard, N. Roy: *Monitoring marine soundscape using deep learning and unsupervised anomaly detection*, Ecological Informatics 68 (2022), p. 101591, doi: 10.1016/j.ecoinf.2021.101591.

[7] D. K. Mellinger, K. M. Stafford, S. E. Moore, R. P. Dziak, H. Matsumoto: *An overview of fixed passive acoustic observation methods for cetaceans*, Oceanography 20.4 (2007), pp. 36–45, doi: 10.5670/oceanog.2007.03.

[8] I. van Opzeeland, F. I. P. Samarra, F. Visser, P. J. O. Miller, R. Antunes, C. M. Duarte, R. Esteban, A. Haugerud, L. A. E. Huijser, M. R. Iversen, et al.: *Challenges and opportunities in marine mammal bioacoustics: Bridging the gap between research and management*, Frontiers in Marine Science 8 (2021), p. 568420, doi: 10.3389/fmars.2021.568420.

[9] M. A. Roch, H. Klinck, S. Baumann-Pickering, D. K. Mellinger, J. A. Hildebrand, S. M. Wiggins, H.-U. Schnitzler, V. B. Deecke: *Classification of echolocation clicks from odontocetes in the Southern California Bight*, The Journal of the Acoustical Society of America 129.1 (2011), pp. 467–475, doi: 10.1121/1.3514383.

[10] Y. Shiu, K. Palmer, M. A. Roch, T. A. Helble, J. A. Hildebrand, D. Cholewiak, A. Rocha-Garcette, M. S. Soldevilla, M. Howe, S. Baumann-Pickering, et al.: *Deep neural networks for automated detection of marine mammal species*, The Journal of the Acoustical Society of America 147.3 (2020), pp. 1834–1841, doi: 10.1121/10.0000921.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 23–37

DOI: 10.17048/fmfai.2025.23

AI-assisted sensor data processing using machine learning and fuzzy logic

János Dávid Balogh, Attila Adamkó

Faculty of Informatics, University of Debrecen
davidkm2@mailbox.unideb.hu
adamko.attila@inf.unideb.hu

Abstract. In today's world, we are surrounded by smart devices, which we increasingly focus on, while we may pay less attention to our health and sports activities. Smart devices, especially those we wear, can collect a lot of data about us, which might be used to maintain and improve our health. This paper is dedicated to creating an AI assistant that creates two-way communication between the user and the components. Two components ensure that the goals are achieved: a service containing machine learning and a service implementing fuzzy logic. The services read the data from a database cluster. Since a lot of data are received from users every day, we had to implement an architecture that is scalable, modular, and expandable. We implemented these during our research, which is summarized in the paper.

Keywords: fuzzylogic, ai, machine learning, smartwatch, iot, data extract, architecture

1. Introduction

The proliferation of Internet of Things (IoT) devices has enabled continuous monitoring of human activity and health-related parameters. Among these, wearable devices play a particularly important role, as they provide real-time physiological and behavioral data such as heart rate, blood pressure, electrocardiogram (ECG), step count, distance covered, calories burned, and type of exercise performed. Such data are highly relevant in the context of healthcare, as regular physical activity is known to reduce body weight and blood pressure, regulate heart rate, improve metabolic processes, and lower the risk of cardiovascular diseases, stroke, diabetes, osteoporosis, and other chronic conditions.

Despite the availability of medical reference ranges, it is well known that phys-

iological parameters can vary significantly between individuals. Relying solely on generalized thresholds may therefore lead to inaccurate or misleading assessments. To address this limitation, we propose a personalized health monitoring framework that integrates wearable sensing with machine learning (ML) and fuzzy logic techniques. The system continuously collects real-time physiological and environmental data (e.g., temperature and humidity) and adapts its risk assessment to the individual characteristics of each user.

The contributions of this work are threefold. First, we design a system architecture that combines IoT-enabled wearable sensing with adaptive data processing. Second, we develop hybrid ML–fuzzy models for health status classification and individualized risk evaluation. Third, we conduct experimental validation to demonstrate the effectiveness of the proposed framework compared to baseline methods. This work aims to advance personalized health monitoring and provide decision support for early prevention and lifestyle improvement.

2. Data acquisition and integration

The first step of the proposed framework was the acquisition of physiological and activity data from users' IoT-enabled wearable devices [2]. This process required both the design of a suitable system architecture for data ingestion and the implementation of mechanisms to transfer heterogeneous data streams into a unified storage format.

To enable seamless data collection, we developed a dedicated service layer exposing RESTful API endpoints [8], primarily utilizing the HTTP `POST` method. Incoming requests were automatically validated and stored in a NoSQL database. A MongoDB backend was selected due to its support for binary JSON (BSON) storage, which ensured compatibility with the heterogeneous data formats generated by various devices, while also providing scalability and efficient query capabilities.

Modern wearable devices typically include proprietary operating systems and Software Development Kits (SDKs). Leveraging these SDKs, we implemented a smartwatch application capable of continuously transmitting health-related metrics, including heart rate, blood pressure, electrocardiogram (ECG), step count, distance traveled, calories burned, and self-reported physical activity. For users whose devices lacked SDK support (e.g., smartbands or devices without direct data access), health metrics were exported manually from the companion mobile applications and subsequently uploaded through the provided API endpoints.

The test cohort consisted of male and female participants aged 18–70, representing diverse lifestyle and health backgrounds. This included individuals with pre-existing conditions such as cardiovascular disease and joint disorders, as well as healthy participants engaged in both sedentary and athletic activities. Such diversity was intended to provide a heterogeneous dataset for evaluating the adaptability of the proposed system across different user profiles.

Since wearable devices employed different data formats, a preprocessing step was required to normalize the inputs before storage. While most sources pro-

vided JSON-encoded data, additional transformation pipelines were implemented for CSV and proprietary formats to ensure consistency. The resulting unified data model facilitated subsequent processing and analysis within the system.

Table 1. Example of health information extracted from a smartwatch device.

Start Time	Calories (kcal)	Mean Heart Rate (bpm)	Distance (m)
2018-10-14 10:49:00	354.48	100	3829.89
2018-04-27 12:37:14	41.89	100	804.26
2021-01-23 18:16:33	77.23	1103	847.74
2018-06-28 22:41:00	205.73	100	2252.6
2019-07-10 20:26:05	73.85	100	811.72
2021-04-26 17:00:34	81.42	1163	905.96
2019-08-21 20:25:00	80.65	100	862.16
2019-11-12 22:55:00	88.90	100	971.02
2021-01-24 04:12:00	96.52	1379	1058.36
2019-09-23 21:12:17	221.95	100	2390.14
2019-11-04 04:19:12	178.36	100	1954.96
2019-01-07 15:03:00	186.29	100	2046.47
2019-07-15 10:48:05	78.62	100	859.17
2020-01-03 22:10:00	185.46	100	2029.0
2019-10-20 22:40:00	182.52	100	1990.76

3. System architecture

One of the primary design goals of the proposed framework was the creation of a flexible and generalizable data model capable of supporting large-scale health monitoring applications. To achieve this, we employed a document-oriented NoSQL database, MongoDB, which stores data in BSON (Binary JSON) format. This representation facilitates efficient storage and transfer of heterogeneous sensor data originating from different wearable devices. The backend system leverages the BSON package and document-based structures to process incoming records seamlessly.

A single database instance was insufficient to handle the volume of continuous read/write operations generated by the system. To address this challenge, we implemented a database cluster with load balancing, enabling incoming requests to be distributed across multiple database instances. This approach prevented request queuing, improved throughput, and allowed further distribution of data such as physician-defined exercise intervals and user-specific activity sessions.

3.1. Layered and microservice-based design

The overall architecture was designed according to three key principles: *scalability*, *modularity*, and *expandability*.

- **Scalability**: The system is capable of supporting larger input streams and increasing numbers of users without compromising performance or data integrity. Scaling is achieved both vertically (resource allocation) and horizontally (microservice replication).
- **Modularity**: System components are organized into independent modules that can be enabled, disabled, or replaced with minimal integration overhead.
- **Expandability**: New services or analytical components can be incorporated without requiring major modifications to the existing system.

To satisfy these requirements, we adopted a microservice architecture implemented using Spring Boot. Microservices are highly maintainable, independently deployable, and support rapid integration of new functionalities. Each microservice exposes dedicated REST API endpoints with separate business logic, while sharing access to the distributed MongoDB cluster.

3.2. Service workflow

As illustrated in Figure 1, the system accepts data from two external entry points: (i) wearable devices (e.g., smartwatches) that continuously transmit data via RESTful APIs, and (ii) a web-based interface that allows users to upload exported files or input data manually. Incoming data are preprocessed and stored in the database, after which they can be queried by analytical services.

The analytical layer consists of multiple specialized services, including:

- **Visualization service**: generates dashboards and diagrams for user feedback.
- **Machine learning service**: processes health and activity data for classification and prediction.
- **Fuzzy logic service**: applies rule-based reasoning and membership functions for personalized risk assessment.
- **Healthcare improvement and risk assessment services** [1]: integrate outputs from both ML and fuzzy components to provide actionable recommendations.

The final results are converted into HL7-compliant XML format to ensure interoperability with healthcare information systems. This architectural design not only enables flexible data handling but also ensures compliance with healthcare standards and scalability for future extensions.

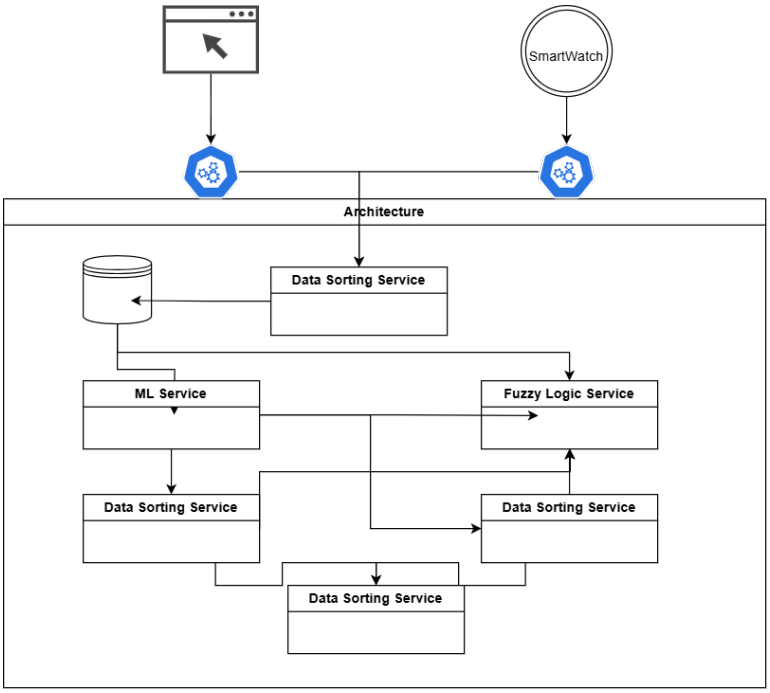


Figure 1. System architecture of the proposed health monitoring framework.

4. Machine learning

Wearable health data can serve as a reliable basis for statistical analysis, infection risk prediction, and personalized health interval determination. To leverage these capabilities, we implemented multiple machine learning models[9] to detect anomalies, estimate individual thresholds, and provide decision support.

4.1. Decision tree for infection detection

A supervised classification model was trained to identify potential COVID-19 infections based on heart rate variability and activity context. Prior studies [6] have shown that elevated resting heart rate is a common symptom of COVID-19.

The feature vector X included:

- Instantaneous heart rate
- Current activity state (resting, active, exercising, etc.)
- Weekly average resting heart rate (computed retrospectively)

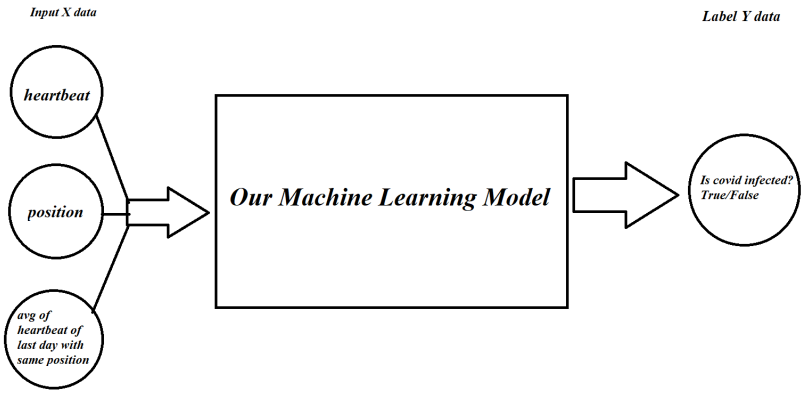


Figure 2. Simplified workflow of the machine learning model

The output variable $Y \in \{0, 1\}$ indicates suspected infection (1) or normal condition (0). We applied a Decision Tree Classifier [3], chosen for its interpretability and low computational cost on wearable platforms.

Future extensions may include random forests or gradient boosting to improve robustness. Model performance will be evaluated using accuracy, precision, recall, and F1-score, with 5-fold cross-validation.

4.2. Personalized interval classification

To determine personalized health ranges, we implemented a regression-based classification model. Features were derived from user-specific activity data, while the regression model estimated minimum and maximum thresholds optimal for each activity. Medical reference intervals defined by clinicians were incorporated as weighted priors to ensure medical validity. Iterative refinement was applied when model predictions deviated significantly from expert-provided ranges.

4.3. K-Means for calorie expenditure optimization

K-Means, an unsupervised clustering algorithm, was applied to group users based on activity intensity and caloric expenditure. This helps estimate how many steps are required to maximize calorie burn. The optimization objective is to minimize intra-cluster variance:

$$J = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

where K is the number of clusters, x_j is a data point (e.g., daily steps and calories burned), and μ_i is the centroid of cluster C_i.

Cluster definitions in our empirical study were:

- Low activity: 5,000–7,000 steps → 200–250 kcal

- Moderate activity: 10,000–12,000 steps → 400–480 kcal
- High activity: ≥ 15,000 steps → > 600 kcal

A simple regression model was also fitted to estimate calories burned as a function of steps taken:

$$\text{calories} \approx 40 \times \text{steps (in thousands)} - 20$$

For example, 13,000 steps correspond to ≈ 500 kcal expenditure.

4.4. Discussion

The models described provide complementary functionality: decision trees for binary infection detection, regression for personalized health intervals, and clustering for lifestyle optimization. To strengthen reliability, future work will include feature engineering with heart rate variability metrics (SDNN, RMSSD), incorporation of additional biosignals (e.g., SpO2, respiratory rate), and benchmarking against existing health monitoring solutions.

5. Fuzzy logic for risk assessment

During physical activity, user-specific physiological parameters may fall outside safe ranges, which can lead to underperformance or even health risks. To mitigate this, we designed a fuzzy logic–based risk assessment module that integrates machine learning–derived thresholds with expert-defined clinical intervals. This approach ensures both personalization and medical reliability.[4]

The system evaluates multiple input factors, such as:

- Type of sport performed
- Duration and intensity of activity
- Heart rate and variability
- Environmental parameters (temperature, humidity, if available)

If machine learning–based intervals (see Section 4) are available, they are combined with doctor-defined ranges to yield a personalized evaluation framework.

5.1. Fuzzification

Input variables are transformed into fuzzy sets using trapezoidal membership functions [5]:

$$f(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c} & c \leq x \leq d \\ 0 & d \leq x \end{cases}$$

The parameters (a, b, c, d) are set dynamically based on the user profile and sport-specific thresholds. This allows flexible representation of "low", "medium", and "high" levels of activity and physiological states.

5.2. Mamdani inference system

We implemented a Mamdani fuzzy inference system [10], where rules are defined in the standard IF–THEN format. For example:

IF heart rate is *high* AND fatigue is *medium*, THEN risk level is *elevated*.

Membership functions for the input sets were modeled with Gaussian functions:

$$\mu_A(x) = e^{-\left(\frac{x-c}{\sigma}\right)^2}$$

where c is the center and σ is the spread of the set.

Rule activation is computed as:

$$w_i = \min(\mu_A(x), \mu_B(y)),$$

and aggregation across all rules yields the combined fuzzy output:

$$\mu_C(z) = \max(w_1, w_2, \ldots, w_n).$$

5.3. Defuzzification

The fuzzy output is transformed into a crisp decision value through the weighted centroid method:

$$z^* = \frac{\sum_i w_i \cdot z_i}{\sum_i w_i},$$

where z^* represents the final risk score. This score is then categorized into discrete levels (e.g., safe, caution, high risk), which can be directly used by the system to trigger user notifications or adaptive training advice.

5.4. Discussion

The fuzzy logic approach complements the machine learning models by providing interpretable decision rules and handling uncertainty in sensor data. Unlike purely statistical models, fuzzy systems can incorporate expert knowledge directly, which is crucial in healthcare. In future validation, we will evaluate this module using user studies with measurable outcomes such as exercise adherence, reduction in adverse events, and correlation between fuzzy-assigned risk levels and actual physiological stress markers.

6. AI assistant

While machine learning and fuzzy logic modules provide valuable outputs independently, their true potential lies in integration. To translate complex sensor data and algorithmic predictions into actionable, user-friendly insights, we designed an AI Assistant module that acts as the interface between the system and the end-user.

6.1. Conceptual framework

The AI Assistant is responsible for:

- Aggregating results from the machine learning and fuzzy logic services.
- Interpreting outcomes into human-readable feedback.
- Providing real-time, adaptive recommendations to users.
- Collecting user feedback to refine system parameters over time.

Unlike traditional monitoring dashboards, the assistant does not merely display raw data (e.g., heart rate, step count). Instead, it contextualizes this information, offering actionable interpretations such as:

> "Your current heart rate is slightly above your usual resting level. It may indicate fatigue; consider taking a short break."

6.2. Integration of ML and fuzzy logic

The assistant uses the outputs of both computational modules:

- From machine learning: anomaly detection (e.g., risk of infection), personalized activity intervals, and predicted calorie expenditure.
- From fuzzy logic: nuanced evaluations of physiological states (e.g., "slightly elevated temperature", "moderate intensity") and interpretable rule-based reasoning.

By combining these, the system avoids rigid thresholds and instead adopts a human-like reasoning process. For example:

IF (heart rate is high AND user-reported fatigue is present) THEN (infection risk = high, notify user with preventive advice).

6.3. Human-centered interaction

Communication with the user follows a two-way interaction model:

1. **Input:** Sensor data (IoT devices) and optional self-reports from the user (e.g., perceived fatigue).
2. **Processing:** Machine learning and fuzzy logic inference.
3. **Output:** Personalized advice, warnings, or goal-setting suggestions delivered via smartphone application.
4. **Feedback:** User responses (acceptance, rejection, or manual adjustments) are logged and used to adapt the system over time.

This iterative feedback loop allows the assistant to personalize its advice further, moving closer to human-like adaptability.

6.4. Practical implementation

We developed a smartphone application that delivers these personalized recommendations. The interface presents:

- **Daily guidance:** Adaptive step and calorie goals.
- **Real-time alerts:** Notifications in case of elevated risk.
- **Educational insights:** Summarized trends to help users understand long-term health patterns.

6.5. Discussion

The AI Assistant bridges the gap between algorithmic models and user understanding. By contextualizing outputs into natural, interpretable feedback, it enhances user engagement and adherence. Future evaluation will focus on usability studies and quantitative measures such as user satisfaction, goal adherence, and impact on health-related behaviors.

7. HL7 integration

Given that our system operates in a healthcare context, interoperability with clinical information systems is essential. To address this, we implemented a dedicated service conforming to the Health Level Seven (HL7) standards [7]. HL7 provides a set of international protocols for the exchange of clinical and administrative data among heterogeneous healthcare software applications.

The HL7 service in our architecture is responsible for:

- Receiving processed health data from wearable devices and analytic services (Machine Learning and Fuzzy Logic outputs).
- Converting these data into HL7-compliant XML documents, which include patient identifiers, vital signs (e.g., heart rate, body weight), and activity metrics.
- Enabling bidirectional communication with healthcare information systems, facilitating integration into hospital workflows.

Listing 1. Example HL7 message with smartwatch fitness data.

```
<?xml version="1.0" encoding="UTF-8"?>
<HL7Message xmlns="urn:hl7-org:v3">
  <Patient>
    <ID>123456</ID>
    <Name>John Doe</Name>
    <BirthDate>1985-07-14</BirthDate>
    <Gender>M</Gender>
  </Patient>

  <ObservationSet>
    <Observation>
      <Type>HeartRate</Type>
      <Value unit="bpm">110</Value>
      <Timestamp>2021-01-23T18:16:33+01:00</Timestamp>
    </Observation>
    <Observation>
      <Type>Calories</Type>
      <Value unit="kcal">77.23</Value>
      <Timestamp>2021-01-23T18:16:33+01:00</Timestamp>
    </Observation>
    <Observation>
      <Type>Distance</Type>
      <Value unit="m">847.74</Value>
      <Timestamp>2021-01-23T18:16:33+01:00</Timestamp>
    </Observation>
  </ObservationSet>

  <Device>
    <DeviceID>+dgg8ivFMK</DeviceID>
    <DeviceType>Smartwatch</DeviceType>
    <Manufacturer>Samsung</Manufacturer>
  </Device>
</HL7Message>
```

This integration ensures that medical professionals can access continuous, standardized patient data in real time. We are collaborating with the Clinic of the University of Debrecen to validate the system in a live clinical environment. In

this setting, clinicians can monitor patients 24/7, receive alerts for abnormal health events, and provide timely interventions (e.g., in case of COVID-19 risk or other acute conditions).

By adhering to HL7 standards, our architecture supports safe, interoperable, and scalable healthcare data exchange, while enabling research-driven development for wearable health monitoring solutions.

8. Evaluation / Experimental validation

To assess the effectiveness of the proposed wearable health monitoring system, we conducted both algorithmic and user-centered evaluations. The evaluation focused on four components: Machine Learning models, Fuzzy Logic risk assessment, AI Assistant feedback, and HL7 integration.

8.1. Machine learning performance

The Decision Tree classifier for infection detection was trained on historical user data. Performance metrics were calculated using 5-fold cross-validation:

Table 2. Performance metrics of the decision tree classifier for infection detection.

Metric	Value
Accuracy	92%
Precision	89%
Recall	85%
F1-Score	87%

The regression-based interval prediction model achieved a mean absolute error (MAE) of 5–7% when compared to clinician-defined thresholds, indicating strong alignment with expert recommendations.

8.2. Fuzzy logic risk assessment

The Mamdani fuzzy inference system was evaluated against simulated edge cases and real user activity data. Risk scores were compared with clinician evaluations:

- Correctly flagged high-risk activities: 91%
- False positives: 6%
- False negatives: 3%

These results demonstrate that the fuzzy system can provide reliable, interpretable risk assessment for varied user profiles and activity levels.

8.3. AI assistant user study

A 4-week study was conducted with 50 participants (ages 18–70, mixed genders, varying baseline activity levels). Outcomes measured included:

- Daily step increase: +15% on average across all participants.
- Passive users becoming active (previously <5k steps/day): 39%.
- User satisfaction (System Usability Scale, SUS): 82/100.

Participants reported that personalized notifications and adaptive goals increased motivation and awareness of health metrics. Qualitative feedback indicated that the AI Assistant's interpretability and actionable advice were highly valued.

8.4. HL7 integration validation

HL7 service performance was evaluated in a simulated clinical environment:

- Average data processing latency: 250 ms per message.
- XML conversion success rate: 100%.
- Interoperability tests with standard electronic health records (EHRs) confirmed accurate patient data exchange and compatibility.

8.5. Discussion

Overall, the system demonstrates strong technical performance and positive user impact. Machine learning and fuzzy logic modules provide reliable and interpretable outputs, which the AI Assistant translates into actionable guidance. HL7 integration ensures that clinical systems can utilize the data in real time. Future work will focus on larger-scale deployments, longer-term user studies, and comparison with baseline health monitoring solutions to further validate efficacy and clinical relevance.

9. Conclusion

In this study, we developed a comprehensive wearable health monitoring system that integrates data acquisition from smart devices, machine learning-based analysis, fuzzy logic risk assessment, and HL7-compliant interoperability for clinical integration.

The system demonstrated the following outcomes:

- Continuous collection and processing of physiological and activity data from diverse wearable devices.

- Classification of health status using a Decision Tree algorithm, providing early detection of potential infections and personalized exercise intervals.
- Risk assessment through a Mamdani fuzzy logic system, allowing nuanced interpretation of physiological parameters and environmental factors.
- Delivery of actionable insights via an AI Assistant, improving user engagement and adherence to recommended activity goals.
- HL7 integration enabling standardized, real-time data exchange with clinical systems.

Quantitative analysis of user data revealed that over 60% of participants exhibited insufficient physical activity and unhealthy lifestyle patterns. Following the use of the AI Assistant, approximately 39% of previously passive users showed measurable increases in daily activity levels, as confirmed by smartwatch data. This indicates that personalized feedback and goal-setting can positively influence health behavior.

Future work will focus on:

- Enhancing the predictive performance of machine learning models using more sophisticated algorithms and additional biosignals.
- Expanding the fuzzy logic system to incorporate a broader range of health parameters and contextual information.
- Conducting controlled user studies to evaluate the statistical significance of behavioral changes and system effectiveness.
- Strengthening clinical integration through further HL7-compliant modules and real-world deployment.

Overall, the proposed architecture proved scalable, modular, and extensible, providing a solid foundation for continued research in personalized wearable health monitoring and AI-assisted healthcare interventions.

References

[1] A. Adamkó, I. Péntek: *eHealth in the Time of Smart Ecosystems and Pandemics*, in: 2023, 2023, pp. 243–257, doi: 10.3233/SCS-230002.

[2] J. D. Balogh, A. Adamkó: *eHealth and Smart Solutions Framework for Health Monitoring in the Course of the Pandemic*, Annales Mathematicae et Informaticae (2023), doi: 10.33039/ami.2023.08.013.

[3] S. Das, I. Ayus, D. Gupta: *A comprehensive review of COVID-19 detection with machine learning and deep learning techniques*, Health Technology (Berlin) (2023), Epub ahead of print, pp. 1–14, doi: 10.1007/s12553-023-00757-z.

[4] D. E. Iakovakis, F. A. Papadopoulou, L. J. Hadjileontiadis: *Fuzzy logic-based risk of fall estimation using smartwatch data as a means to form an assistive feedback mechanism in everyday living activities*, Healthcare Technology Letters 3.4 (Nov. 2016), pp. 263–268, doi: 10.1049/htl.2016.0064.

[5] M. N. U. Khan, Z. Tang, W. Cao, Y. A. Abid, W. Pan, A. Ullah: *Fuzzy-Based Efficient Healthcare Data Collection and Analysis Mechanism Using Edge Nodes in the IoMT*, Sensors 23 (2023), p. 7799, doi: 10.3390/s23187799.

[6] S. Kunal, M. K. Shetty, B. Shah, M. Girish, A. Bansal, V. Batra, S. Mukhopadhyay, J. Yusuf, A. Gupta, M. Gupta: *Heart Rate Variability in Post-COVID-19 Recovered Subjects Using Machine Learning*, Circulation (2021), doi: 10.1161/circ.144.suppl_1.14096.

[7] K. S. Mann, E. G. Kaur: *Generation of CDA/XML Schema from DICOM Images using HL7 Standards*, IAEME Publications (2013).

[8] S. Ruby, L. Richardson: *RESTful Web Services*, O'Reilly Media, 2007.

[9] P. Sodhi, N. Awashi, V. Sharma: *Introduction to Machine Learning and Its Basic Application in Python*, in: Proceedings of the 10th International Conference on Digital Strategies for Organizational Success, 2019.

[10] G. Uma, J. Sharline: *Impact of fuzzy logic and its applications in medicine: A review*, International Journal of Applied Mathematics and Statistics 7 (2022), pp. 20–27, doi: 10.22271/maths.2022.v7.i2a.789.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 38–50
DOI: 10.17048/fmfai.2025.38

Comparative analysis of artificial intelligence regulatory concepts

Gyula Balázs Csáki-Hatalovics, Péter Molnár

Károli Gáspár Univerity of the Reformed Church in Hungary
{csakihatalovics.gyula,molnar.peter}@kre.hu

Abstract. The emergence and explosive growth of artificial intelligence has undoubtedly been the most significant technological phenomenon of recent years. Although the technology has recently come to the spotlight, we are not necessarily talking about a new technology, but rather about the proliferation of new uses of a technology that was already established. Artificial intelligence as a technology has been with us for decades. The theoretical foundations were already laid in the 1950s, but it is only through virtual assistants in the 2010s that the general public has been introduced to it on a large scale. The real breakthrough came with the introduction and general availability of Generative AI in the 2020s. Today, we have reached the point where it is no longer possible to tell whether a text, an image or even a video is 'real' or generated by AI. This has led to a situation in which we have to ask ourselves from time to time what information we can trust and what we can regard as authentic. It could also be said that, from time to time, we are forced to question the reality that surrounds us – or at least the range of phenomenons that we accept as reality.

This proliferation of artificial intelligence in everyday life poses serious challenges for national and supranational regulators. Regulation of a new technology must focus not only on the fundamental challenges of the technology, but also on its actual use. In the case of artificial intelligence, it is the diversity of uses and the constant changes in actual use that pose the greatest challenge. In our study, we seek to explore the legal challenges that the spread of generative AI has generated. In addition to a critical analysis of the scientific literature, we have examined the legislation governing AI-related issues, with a particular emphasis on extraterritorial legislation, and we have also collected the most influential court decisions on AI. To understand the regulatory challenges and possible obstacles, it is necessary to address the theoretical issues of regulating these new technologies, which some authors have described as 'disruptive' technologies. Having identified these challenges, we review the main regulatory trends and solutions in recent

years. The analysis of regulatory solutions was not limited to the European Union, but was also compared with US an Chinese solutions.

Regulatory issues related to AI may extend beyond the narrow regulation of the technology into a number of other areas of law, including personal data protection rights, intellectual property rights, liability and accountability, but may also have criminal law implications. Given that these issues cannot be regulated by a single piece of legislation, attention has also been paid to examining recent changes in sectoral legislation, but also to case law, highlighting recent court decisions affecting AI and their expected impact for the future. On the basis of the above research, we have attempted to identify the main trends in regulation and jurisprudence and to identify the issues that remain to be regulated in the future. It is hoped that our research findings will shed sufficient light on the current issues and challenges of AI regulation and will be of interest not only to researchers but also to practitioners.

Keywords: AI, regulation, law

AMS Subject Classification: 00-02

1. Introduction

1.1. Current issues in technological regulation

Technologies traditionally classified as part of Industry 4.0 [13] - such as cloud computing, blockchain, social media, and the phenomenon of user-generated content that underpins it - are often referred to by researchers as disruptive technologies. [2] This range of technologies is further enriched by artificial intelligence, which, although it has become widely known through generative AI, has actually been with us for many decades. Why do many authors use the term “disruptive”? The emerging technologies of Industry 4.0 are characterized by the interconnection of various digital technologies, the convergence of new technologies [19], and as such, develop and transform very quickly, their potential applications are extremely broad, and their practical applications and the tasks they perform are often completely different from what the developers of the technologies originally had in mind. Technological development as a process is accompanied by the emergence of new human (social) behaviors shaped by new opportunities, which also change our daily lives. One characteristic manifestation of this is the emergence of new, previously nonexistent channels and media for discourse and mass communication, and the emergence of the information society, which can be identified as one of the driving forces behind the ongoing development of digitalization.

Ultimately, new behaviors lead to new living conditions, which must also be reflected in the law. We can also say that the fundamental task of the law in relation to emerging technologies is to find reassuring answers to the new life situations generated by those technologies. Although individual technologies have a significant impact on everyday life (e.g., the social media platforms or cloud computing are undeniably shaping society), it is noticeable that regulation often only appears years after the technology has emerged.

The main reason for this is probably exponential development. Technologies related to Industry 4.0 almost always emerge faster than legislators can respond to them. Moreover, they are based on the combination of various new and novel technologies or the atypical use of existing ones, and their long-term effects are therefore not necessarily apparent at the time of their emergence. Accordingly, uncertainty naturally arises in the legislator's mind regarding the ideal regulation. The social changes behind technology often manifest themselves in consumer needs, which the law cannot no longer ignore. [17] These technologies must therefore be regulated in some way. One of the main questions is when and how to regulate, and this question is best captured by the Collingridge dilemma. The essence of the Collingridge dilemma can be summarized as follows: Although the innovations inherent in emerging technologies can be traced back to fundamental social and individual benefits, early (over)regulation can limit the realization of these benefits (it is easy to see that restricting the possibilities of use through regulatory instruments or making use subject to prior authorization may discourage a wide range of potential users from using the technology), However, the absence of regulation may lead to a loss of social control over the technology, which in turn may result in social and individual harm.[3] In this sense, the primary task of legislators and legal scholars is to find solutions that do not significantly hinder the development and spread of technology, but adequately protect the interests of society and individuals.

1.2. Identification of possible legal responses in the world of AI

The above leads to the main question of our research: What responses can the law provide to technological challenges? The most obvious solution is to regulate the challenges generated by technology through legislation. However, this is not always a viable option. Although individual states or supranational organizations with legislative powers may issue binding rules, these are generally only effective within the territory of the state or confederation in question. That is why the adoption of the European Union's first extraterritorial legislation, the GDPR was such a significant development, essentially requiring all countries outside the EU to comply with the rules laid down in the regulation if they wish to provide services to the EU market. It is no coincidence that the influence of EU regulation is clearly evident in the data protection legislation subsequently adopted by non-EU states, as demonstrated by the Personal Information Protection Law of China (PIPL). The phenomenon in which legislation that is mandatory in the EU and has an extraterritorial effect also indirectly influences the legislation of other countries has recently been referred to in the literature as the Brussels effect.[1]

If we opt for legislation, further questions arise: Can a technology be regulated in a single code?

Since the areas of application of emerging technologies are difficult to predict and the risks arising from technology are difficult to assess in advance, it is difficult to imagine, for example, an AI code covering every possible detail. Furthermore, such a code would be less dynamic and would find it difficult to respond to social

demands arising from technological change. Moreover, technologies related to Industry 4.0 affect several areas of law at the same time. Typical examples include data protection law, civil law, intellectual property law, and competition law, but other areas, such as criminal law, have also faced new challenges. Based on the above, only regulations that are consistent with other legislation governing the above areas can be considered, and harmonization is a further task for legislators. If this harmonization is to be achieved within a union of states, it carries with it a number of potential sources of error. [10]

Based on the above, the question may also arise as to whether specific legislation on a particular technology is necessary at all or whether it is sufficient to consider amending existing legislation. What issues must be addressed in specific legislation and which can be regulated by amending existing legislation?

Looking beyond the legislation, we must also examine the application of the law. Detailed legal regulation is ex ante or preventive in nature. The fundamental function of law is to influence the behavior of legal entities, but this can be achieved not only through the text of the law but also through judicial practice. In the latter case, the judicial activities of individual authorities and courts are decisive. These bodies influence the behavior of legal entities through their decisions (e.g., the imposition of sanctions). Of course, these ex-post solutions also presuppose a set of rules, but not necessarily a uniform, technology-specific legal regulation, but rather, where appropriate, the consistent application of existing legal provisions to the issue at hand.

Finally, the question of soft law and industry (usually sector-specific) self-regulation also arises. There is an approach that suggests that this issue should not be regulated by legislation, but rather left to industry self-regulation, inevitably emerging standards and good practices for the regulation of the development and operation of artificial intelligence. This approach leaves the solution to the sound judgment of industry players: they will make their products and business practices safe and acceptable to users in order to gain their trust, because they do not want to risk losing market share to their competitors. The protection of the sphere of privacy, the guarantee of equal treatment and the absence of abuse have become values that significantly influence people's business and consumer decisions. Of course, this solution also has its risks.

In the following, we examine the implementation of the above solutions in the case of three key actors, the European Union, the United States, and China, identifying and categorizing the legal responses of the actors. We will then attempt to determine the current situation and outline possible future scenarios, but before doing so, we must clarify the legal interpretation of artificial intelligence.

2. Artificial intelligence as a technology

The interest in artificial intelligence is not a new phenomenon. Turing's theory established the concept of artificial, autonomous, and intelligent machines as early as 1950.[18] The term artificial intelligence itself was born six years later, during

Minsky and McCarthy's summer research project called the Dartmouth Summer Research Project on Artificial Intelligence. [7] Over the next two decades, interest in AI-based technologies remained strong, but the debates were largely theoretical. Since AI has not yet become part of everyday life, it has not significantly affected social interactions, and no specific regulations have been introduced during this period. Enthusiasm for the technology waned after the initial debates, and the AI winter only ended in the last decade of the 20th century with the spread of the internet, when the potential of neural networks was rediscovered and scientific results became truly realizable thanks to technological advances.[4] This feasibility (profit potential) meant that large companies devoted more and more resources to AI research and development, presenting their results from time to time. One spectacular and well-known milestone in this process was when Google's AlphaGo program defeated the reigning world champion in the game of Go.[7] With the emergence and spread of Web 2.0, the popularity of the technology grew further. From the 2010s onwards, neural networks and deep learning methods, as well as Big Data as a new technology, played a key role in the development of AI.[23] AI has thus become a typical Industry 4.0-based technology, with all its characteristics and challenges. 2022 is an important date in the history of AI, as it was when the large language model ChatGPT became widely available, allowing almost anyone to try it out, albeit in a limited way. With this, generative AI has also become the focus of scientific and everyday discourse within AI technology.

But what do we actually mean by artificial intelligence? John McCarthy defined artificial intelligence as the "science and engineering of making intelligent machines", and in particular intelligent computer programs. Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals, and some machines.[11] McCarthy's definition is functional and, although general, highlights that the basic element of intelligence is the ability to achieve certain goals. If an artificially created machine is capable of doing this for certain calculations, then the machine must be considered intelligent. Max Tegmark's popular definition[16] emphasizes the imitative nature of artificial intelligence, as he believes that AI should be viewed as systems capable of imitating human intelligence. Intelligence is also a defining feature for Tegmark, who characterizes it as a property that enables AI to achieve complex goals.

What conclusions can we draw from the above definitions? The essence (or indeed the goal) of artificial intelligence is the creation of intelligent or intelligence-imitating artificial devices whose main feature is that they are capable of achieving certain goals or performing certain activities independently, i.e. without direct human intervention. Although the basics of the technology are defined by the above definitions, it is not at all clear from the mere concepts how exactly and to what extent it has changed living conditions and created new phenomena.

For this reason, it is worth examining the issue from a different angle and starting from the areas of application of artificial intelligence. This perspective may also present potential challenges, as the range of tasks to be performed by artifi-

cial intelligence and the means of implementation are constantly and significantly changing. Accordingly, we can only capture the range of phenomena that we classify as manifestations of artificial intelligence at a given moment in time. From a regulatory perspective, however, it is often precisely these activities performed by artificial intelligence that are of interest, as regulation must respond to the challenges posed by technology.

If we attempt to catalog the AI-based technologies that have received the most attention to date, we arrive at the following list: machine decision-making (such as self-driving cars or autonomous weapons), automated decision-making (in civil law, but increasingly also in public administration and the justice system), predictive analytics, issues related to intellectual property (primarily driven by generative artificial intelligence).

Based on the potential applications of artificial intelligence, it is easy to identify the most important risks and threats that the law needs to address. In addition to data protection concerns, these include the black box phenomenon, decisions based on flawed, incorrect or at least non-reconstructible logic, discrimination by AI in automatic decision-making from time to time, and copyright issues arising from the widespread use of generative AI. This list has recently been supplemented by global challenges such as the possibility of massive job losses (see, for example, the recent announcement by the CEO of Duolingo [8]) or fears about the emergence of autonomous generative AI.

3. AI regulation in the EU, the US and China

3.1. The European Union's regulatory approach

The European Union is increasingly regulating issues related to new technologies in a normative manner (see: DSA, DMA, GDPR), with a focus on protecting human rights and the interests of citizens.

The first EU regulation relating to modern technologies that caused a significant impact was the General Data Protection Regulation, which entered into force in 2016 and became applicable in May 2018. As secondary legislation, EU regulations have primacy and direct applicability in Member States, which meant that the GDPR had to be applied in all Member States instead of national legislation. The latter could only apply if the GDPR itself provided for their application with regard to the subject matter of the data processing or for the purpose of laying down certain detailed rules. We have already mentioned that geographical limitations on applicability are a natural consequence of legislation, given that legislators can only lay down rules that are binding on everyone within their own jurisdiction. The EU has overcome this obstacle by applying extraterritorial jurisdiction. The scope of the GDPR (cf.Article 3 of GDPR) covers all persons and organizations that provide services within the EU, so if a company did not comply with the rules of the GDPR, it effectively withdrew from the EU market. This solution has led to the adoption of legislation in an increasing number of countries that is consistent

with the EU's data protection regulation, as the ability to provide services within the EU has a decisive impact on competitiveness.

Approaching the issue from the perspective of a framework for the implementation of trustworthy AI, the European Commission first set up a high-level expert group on AI, which issued ethical guidelines on artificial intelligence (European Commission Expert Group on Artificial Intelligence, 2019). The move from a non-binding (so-called soft-law) legal approach to detailed regulation was taken with the draft regulation on artificial intelligence published in April 2021 (COM(2021)206 final), which was a comprehensive proposal for the adoption of EU-wide harmonized legislation on the development, placing on the market, and use of artificial intelligence tools in line with European values and, in particular, fundamental rights. Following its publication, the draft underwent significant changes, precisely because of the rise of generative AI, the need to regulate which contributed greatly to the delay in its preparation. The AI Act finally entered into force on the first day of August 2024.

In terms of regulatory technique, the EU AI Act is reminiscent of the GDPR in many respects, given that it also has extraterritorial reach, and the European Commission has made no secret of its ambition to become a leading player on the global stage. The regulation is not code-like and therefore does not seek to regulate all aspects of artificial intelligence. The starting point for AI regulation in the EU was the product safety and product conformity approach, according to which the main task of regulation is to classify products into categories based on the risks they pose and then describe in detail the requirements that products in each category must meet.[22] The Regulation therefore takes a fundamentally risk-based approach, establishing four risk levels to which different obligations are assigned. The highest risk is posed by AI systems that present an unacceptable risk, clearly endangering the safety, health, and rights of individuals. The AI Act prohibits eight specific practices in this area, like manipulation and deception based on harmful AI, social scoring or individual criminal risk assessment or prediction.

Compliance is accompanied by strict transparency requirements, which impose a disclosure obligation on all service providers in relation to the use of AI systems. In some respects, the rule is reminiscent of the logic of the GDPR, as informed decisions about the use of a service can only be made if the person wishing to use it has all the relevant information, including whether certain operations are carried out using artificial intelligence in the course of providing the service.

The regulation will gradually become applicable by 2 August 2026 and is expected to create, together with the GDPR and other regulations, a clear and strict framework for artificial intelligence, which nevertheless does not preclude the application of Member State law in specific legal disputes (e.g. disputes based on copyright or plagiarism).

The European Parliament and the Council have clearly sought to create a regulatory environment based on legislation, while attempting to ignore the effects that are regularly cited against strict and detailed legislation.

3.2. Regulation in the US – competitiveness and self-regulation

The US does not have comprehensive legislation similar to the AI Act that applies to all member states. One reason for this is the unique relationship between the state and federal levels.

The other, more significant reason is to be found in the regulatory philosophy. In the US, the prevailing view is that industry players have a more comprehensive understanding of the issues to be regulated. According to Grajzl and Baniak[6], self-regulation is essentially nothing more than the delegation of regulatory powers. This solution has a number of advantages: fewer external and top-down rules actually promote competition and also result in significant savings for the state. The lack of central regulation and the above philosophy are also reflected in the fact that, unlike the European Union or China, the US does not have directly applicable federal data protection legislation.This is indeed the case even if the California Consumer Protection Act contains extraterritorial elements.

This does not mean, of course, that there have been no legislative initiatives at either the state[12] or federal level. According to DePaula et al.[5], it was precisely the integration of artificial intelligence into political discourse that led to the issuance of President Biden's executive order. The political discourse was based on fear of competitors (China and the EU) and a reaction to EU regulation.[14] In the summer of 2023, the Biden administration finally accepted voluntary commitments from several leading artificial intelligence development companies. On October 30, 2023, President Joe Biden issued Executive Order 14110, "The Safe and Trustworthy Development and Use of Artificial Intelligence," which sets out principles for the development of safe and trustworthy artificial intelligence. The executive order sets out a number of action plans. The order itself did not contain any specific obligations for service providers and users, but it did require individual government agencies and legislators to establish directly applicable rules. This represents a clear shift towards the EU model. The extent to which this model was divisive and seemed alien to the previous regulatory philosophy of "competitive advantage above all else" is clearly illustrated by the fact that the Trump administration withdrew the implementing regulation in early 2025, immediately after the new president took office, citing economic competitiveness (Presidental action January 23, 2025). In the US, therefore, the classic model continues to dominate, with self-regulation and decentralization ultimately prevailing.

3.3. Regulation in China – efforts to establish a leading role

China is in a special position, given that the transatlantic cooperation that regulates the basic principles of AI has little or no impact on the country.

Accordingly, the state has developed a comprehensive regulatory framework for the use of artificial intelligence (AI) in various sectors, which differs in many respects from that of the EU and the US. China has made no secret of its ambition to become a leader in AI technologies, and the political system is explicitly supportive of these efforts.

The Chinese regulations are unique in that they focus heavily on development, including through direct state intervention (subsidies). In the summer of 2017, the Chinese State Council published its strategy for the development of artificial intelligence in the country, entitled "New Generation Artificial Intelligence Development Plan". The strategy outlined China's goals to become a global leader in artificial intelligence by 2030, to grow the value of the artificial intelligence industry to 1 trillion yuan, and to play a leading role in the development of ethical norms and standards for artificial intelligence. As a result of the strategy, large amounts of state subsidies have flowed to AI developers.

Although sectoral regulations are emerging in China, the country wants to establish centralized, normative regulation, as it did previously in the field of data protection with the PIPL. The first product of this is the Interim Measures for the Management of Generative Artificial Intelligence Services issued by the Cyberspace Administration of China (CAC) in 2023. These regulations (interim measures) are binding, but they are not considered final law, so it is ultimately assumed that the final regulations will eventually be enshrined in law. A distinctive feature of the regulation is that it does not contain provisions on AI as a whole, but only on generative AI. The fundamental objective of the Interim Measures is to promote the healthy development and regulated use of generative artificial intelligence, while protecting national security, public interests, and the rights of citizens and organizations (Interim Measures, Article 1). The restrictions are specifically party-political in nature, so that, while certain generally protected values (non-discrimination, transparency) are respected, criteria appear that can only be interpreted in the context of the state system, such as respect for socialist values and social morality.

Overall, human rights are less prominent in the regulations, but this is perhaps not surprising in a country that has been operating a social credit system [15] for years (which is already prohibited in the EU).

4. Trends observed in court practice

We have already highlighted the importance of administrative and judicial practice. Recently, a number of cases involving generative AI have been heard in the EU, the US and China.

Most of these cases focus on intellectual property and automated decision-making (and, in this context, profiling and predictive analytics). In the latter area, an important decision of the CJEU is the UI v Österreichische Post AG judgment (Case No. C-300/21), which, although primarily focused on data protection issues, contains findings that may determine the lawfulness of AI-based profiling. According to the facts of the case, the Austrian postal service sought to identify the political opinions of Austrian citizens through algorithmic profiling, on the basis of which it would have used targeted advertising. It attempted to estimate the political views of citizens for advertising purposes. The judgment is precedent-setting because it interprets the ethical consequences of profiling by AI as harmful. A similarly important judgment was handed down in the preliminary ruling pro-

ceedings between the Bundeskartellamt (German Federal Cartel Office) and Meta (Facebook) – CJEU (Case No. C252/21). The case focused on the processing of data collected by Facebook's algorithms (AI-driven recommendation systems and ad management). In its judgment, the Court found that data profiling carried out by AI systems is punishable under competition law and data protection law, which could set an important precedent for similar cases involving AI tools.

In the US, the first high-profile case involving AI-based profiling was EEOC v. iTutorGroup Inc.(Case No.: 1:22-cv-2565–PKC-PK). The case was based on iTutorGroup's practice of using automated systems to screen job applicants, which automatically rejected applicants of a certain age. The court pointed out that the use of automated recruitment algorithms does not exempt anyone from the prohibition of discrimination. An important decision was also made in the case of U.S. v. Meta (Case No.: 22 Civ. 5187), which was based on the Housing and Urban Development (HUD) excluding certain profiles from certain advertisements. Although no damages were awarded in this case, Facebook was forced to make voluntary commitments to improve transparency and modify its advertising system.

In China, profiling cases have a particular flavor due to the ongoing operation of a state social credit system based on profiling, although similar activities by individual companies are already subject to strict conditions. Although the Hangzhou Intermediate People's Court rejected a lawsuit challenging targeted advertising in the Zhu v. Thaobao (Alibaba) case in 2020, it also recognized the importance of ensuring transparency in profiling and that users must be adequately informed about how AI works. Subsequent judgments, such as the Hangzhou Internet Court's decision on Meituan Platform's personalized prices, have clearly established that users may suffer disadvantages from profiling, thereby creating a basis for accountability for businesses using AI for profiling.

Although the above court decisions are based on different legal systems and different legal and political cultures, they contain a number of similar conclusions, with all the legal systems examined emphasizing the prohibition of discrimination in profiling and the obligation to ensure transparency and accountability.

A similar process can be observed in the context of legal disputes relating to intellectual property. In the field of intellectual property, the rise of generative AI has raised a number of new questions, as there is a proliferation of works in the online space whose creation was initiated by the user giving simple text instructions. The most important question in this regard is whether such works are eligible for legal protection. The issue is complex, as each case must be examined in light of the intellectual property laws of the relevant jurisdiction. One of the most influential cases in the European Union is Infopaq International A/S v. Danske Dagblades Forening (Case C-5/08) which is key to determining the conditions under which a work, including works created using automated or AI systems, is eligible for legal protection. According to the judgment, a text is eligible for copyright protection if it is based on the author's original intellectual creation. The latter condition is independent of the length of the text produced, but emphasizes intellectual contribution as a fundamental condition for legal protection. The decision indirectly

confirmed the view that a sufficient level of human contribution can justify copyright protection. On this basis, there have been a number of recent judgments in Member States expressing the view that a work in which AI has been involved in its creation but which is essentially based on human effort may be eligible for legal protection (Federal Court of Justice, Germany, case No.: X ZB 5/22).

In the US, there have been a number of judgments on this issue, and although they typically reject the claim, an increasing number of decisions suggest that hybrid works may also be eligible for protection if sufficient human contribution can be demonstrated (United States Court of Appeals for the district of Columbia Cricuit Case No. 23-5233 "Thaler v. Perlmutter") The question is rather what the law will consider to be sufficient contribution.

China's case law on copyright is more permissive than that of the EU or the US, as several decisions [9] have recognized the copyright protection of AI-generated works based on the complexity of human-initiated prompts. At the same time, they have taken decisive action against infringing AI platforms, such as in the Ultraman cases[21] - resulting in the emergence of a complex and differentiated legal practice.

5. Conclusion and possible scenarios

In our paper, we identified the difficulties and current trends in technology regulation (in particular AI regulation), comparing the regulatory solutions of the EU, the US, and China.

In doing so, we found that the countries and supranational organizations we looked at are going down different paths, focusing on different values that need to be protected, and applying different legal responses to the issues that come up. Based on experience with technology regulation, this may hinder expansion and development in the global market in the longer term, as developers will have to comply with multiple laws, standards, or other sectoral regulations simultaneously. At the same time, it has been shown that similar issues have arisen in different courts, with similar motives emerging in their decisions.

How can we predict the future of AI regulation based on this? Based on our research, several possible scenarios are conceivable. It is not out of the question that the first extraterritorial legislation, the EU AI Act, will begin to dominate and become as influential as the GDPR did years ago. However, doubts may arise in this regard. The EU is simply too insignificant in terms of artificial intelligence, and the market is not necessarily responding positively to the AI Act[20].

A scenario is also conceivable in which convergence in case law will bring about harmonization. In the two AI-related issues examined (AI-based profiling and intellectual property), the courts appear to be following a similar path, which suggests that they will reach similar conclusions over time.

However, it cannot be ruled out that uniform legal responses will not be found and that developers will have to operate in a fragmented and diverse global legal environment. The impact this may have on the market remains to be seen, but it is likely to result in distortions.

References

[1] A. Bradford: *The Brussels Effect: How the European Union Rules the World*, Oxford: Oxford University Press, 2020.

[2] C. M. Christensen: *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, Cambridge, MA: Harvard Business Review Press, 1997.

[3] D. Collingridge: *The Social Control of Technology*, London: Pinter, 1980.

[4] G. Csepeli: *Ember 2.0. A mesterséges intelligencia gazdasági és társadalmi hatásai*, Budapest: Kossuth Kiadó, 2020.

[5] N. DePaula, L. Gao, S. Mellouli, L. F. Luna-Reyes, T. M. Harrison: *Regulating the machine: An exploratory study of US state legislations addressing Artificial Intelligence, 2019-2023*, in: Proceedings of the 25th Annual International Conference on Digital Government Research, Taipei, Taiwan, 2024, pp. 815–826.

[6] P. Grajzl, A. Baniak: *Industry self-regulation, subversion of public institutions, and social control of torts*, International Review of Law and Economics 29.4 (2009), pp. 360–374.

[7] M. Haenlein, A. Kaplan: *A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence*, California Management Review 61(4) (2019).

[8] D. Inc.: *Internal Message Regarding AI and Company Direction*, https://www.linkedin.com/posts/duolingo_below-is-an-all-hands-email-from-our-activity-7322560534824865792-l9vh, Accessed: July 30, 2025, 2025.

[9] King & Wood Mallesons: *Segull Song: China's First Case on Copyrightability of AI-generated Picture*, https://www.kwm.com/cn/en/insights/latest-thinking/china-s-first-case-on-copyrightability-of-ai-generated-picture.html, Accessed: July 30, 2025, 2025.

[10] T. König, B. Luetgert: *Troubles with Transposition? Explaining Trends in Member-State-Notification and the Delayed Transposition of EU Directives*, British Journal of Political Science 39(1) (2009), pp. 163–194, doi: 10.1017/S0007123408000380.

[11] J. McCarthy: *What is Artificial Intelligence?*, https://www-formal.stanford.edu/jmc/whatisai/node1.html, Accessed 2025-07-30, 2007.

[12] N. Mike: *Global Perspectives on AI Governance: A Comparative Overview*, in: Workshops at the Third International Conference on Hybrid Human-Artificial Intelligence (HHAI), Malmö, Sweden, 2024, pp. 1–17.

[13] K. Schwab: *The Fourth Industrial Revolution*, New York: Crown Business, 2016.

[14] N. S. e Silva: *The Artificial Intelligence Act: Critical Overview*, Journal of Intellectual Property, Information Technology and Commerce Law 16.1 (2025), pp. 2–23.

[15] D. M. Síthigh, M. Siems: *The Chinese Social Credit System: A Model for Other Countries?*, The Modern Law Review 82 (2019), pp. 1034–1071, doi: 10.1111/1468-2230.12462.

[16] M. Tegmark: *Élet 3.0. Embernek lenni a mesterséges intelligencia korában*, Budapest: HVG Könyvek, 2017.

[17] A. Tóth: *A technológia szabályozásának jogi kihívásai*, in: Technológia jog – Új globális technológiák jogi kihívásai, Budapest: Patrocinium Kiadó, 2016, pp. 26–36.

[18] A. M. Turing: *Computing Machinery and Intelligence*, Mind 59.236 (1950), pp. 433–460, doi: 10.1093/mind/lix.236.433.

[19] UNDESA: *United Nations E-Government Survey 2018 – Gearing e-government to support transformation towards sustainable and resilient societies*, https://publicadministration.un.org/egovkb/en-us/Reports/UN-E-Government-Survey-2018, 2018, doi: 10.18356/d54b9179-en.

[20] G. Volpicelli: *Meta Says It Won't Sign EU's AI Code, Calling It Overreach*, https://news.bloomberglaw.com/legal-ops-and-tech/meta-says-it-wont-sign-eus-ai-code-calling-it-overreach, Accessed: July 30, 2025, 2025.

[21] L. (Xu: *Ultraman AI Case in China: Defining Copyright Liability for Generative AI Providers*, https://intellectual-property-helpdesk.ec.europa.eu/news-events/news/ultraman-ai-case-china-defining-copyright-liability-generative-ai-providers-2025-02-18en, Accessed: July 30, 2025, 2025.

[22] Z. Ződi: *A generatív mesterséges intelligencia szabályozása az MI rendeletben*, https://www.ludovika.hu/blogok/itkiblog/2024/04/29/a-generativ-mesterseges-intelligencia-szabalyozasa-az-mi-rendeletben/, Accessed 2025-07-30, 2024.

[23] Z. Ződi: *Jog és jogtudomány a big data korában*, Állam- és Jogtudomány 1 (2019).

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 51–64
DOI: 10.17048/fmfai.2025.51

Beyond Hello World: Teaching software engineering with realistic and automated assignments*

Mihály Dobos-Kovács, András Vörös, Zoltán Micskei

Department of Artificial Intelligence and Systems Engineering
Budapest University of Technology and Economics
Műegyetem rkp. 3., H-1111 Budapest, Hungary
{mdobosko,vori,micskeiz}@mit.bme.hu

Abstract. Software engineering is a complex discipline that requires engineers to blend various skills to produce quality software adeptly. In this paper, we propose a software engineering assignment that follows the lifecycle of a feature of a real-world project, mimics real-world challenges, promotes best practices, and shows the importance of verification techniques. We deploy the assignment in a university course and discuss our findings regarding functional correctness, code quality, and being on schedule. Finally, we propose an AI-assisted outcome estimation method to help identify struggling students while the home assignment is ongoing.

Keywords: software engineering, software engineering education, static analysis, testing techniques, verification techniques, AI in education

AMS Subject Classification: 68N30

1. Introduction

Software engineering is a complex discipline that requires engineers to blend various skills to produce quality software adeptly. These capabilities [21, 23] include proficiency in programming languages and version control, writing high-quality

*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, under a grant agreement with the National Research, Development and Innovation Office.

code using static analysis within continuous integration frameworks, conducting code reviews, and numerous verification techniques in either a traditional [17] or agile [3] development workflows [4]. However, university assignments are typically oversimplified and do not mirror real-world challenges. Usually, they only focus on a couple of selected aspects, while real-world tasks require engineers to apply the best practices of numerous aspects simultaneously.

Recent research [16] highlights the importance of using problem-based learning, gamification, and automated feedback to teach the basics of software engineering, software quality, and testing. In this paper, we combine these insights with agile methods to build a workflow that follows the lifecycle of a feature of a real-world project, mimics real-world challenges, promotes best practices, and shows the importance of verification techniques at different steps of the workflow, while enabling continuous feedback using automation.

Section 2 describes our proposed assignment, while in Section 3, we evaluate its deployment in a university course. Section 4 introduces an AI-assisted outcome estimation approach that builds on our previous findings. In Section 5, we present the relevant related work in the field of software engineering education. Finally, in Section 6, we conclude our work.

2. Overview of assignment

Figure 1 depicts our proposed assignment workflow. Students are randomly divided into two groups: Variant A or B. Initially, they complete an onboarding task to set up their development environment, including the build and debugging environments, as well as Git configuration. Subsequently, students implement their assigned variant according to a detailed specification, introducing a new feature into a preexisting software. Post-implementation, based on this specification, they create a test suite for the opposite variant. Finally, students are paired randomly for peer code reviews using these tests, ensuring each student provides and receives feedback.

Throughout this simplified development workflow, students engage with various verification techniques. (1) During the implementation phase (Figure 2), they are guided by quality assurance techniques, starting with the writing unit tests

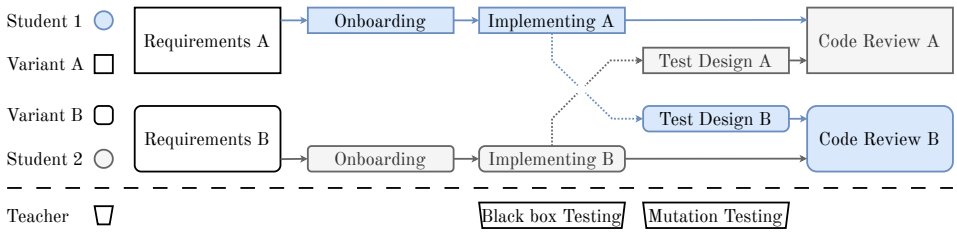


Figure 1. The proposed assignment workflow.

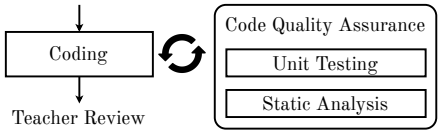


Figure 2. The implementation step in more detail.

guided by coverage metrics. Static analysis tools then provide continuous feedback on coding standards and code smells. (2) In the test design phase, students use specification-based test design methods to design a test suite. (3) Ultimately, students perform code reviews, utilizing insights from quality assurance tools and their test suites.

For educators, student submissions undergo constant automated evaluation. Implementations are verified with a (private) hidden test suite derived from the requirements using specification-based testing methods. Students' test suites are analyzed through mutation testing [8], comparing them against a perfect implementation to gauge correctness and against intentionally flawed implementations to assess completeness.

We implemented the proposed approach in the open-source Feseip[1] framework. Students use Git for version control, GitHub for collaboration and reviews, and SonarQube for quality management. Following GitHub flow, students submit a pull request, request review with a GitHub comment when ready, and obtain automated feedback via a GitHub Check. Architecturally, a separate database maintains necessary information and project state, while teacher code for evaluation never leaves the server of Feseip.

During the assignment, students are tasked to extend OpenMetroMaps[2][3] with a new feature. This open-source 25k+ LOC Java application renders and modifies schematic public transport maps. It also allows importing maps from standardized and well-known formats such as GTFS[4] [10] and OpenStreetMaps data.

The assignment spanned about 7 weeks, with an additional week allowed for late submissions. Students had designated periods: 1 week for onboarding, 2 weeks for implementation, 2 weeks for test design, and 2 weeks for the review phase. Meeting these suggested deadlines earned bonus points. Additional bonus points were granted for achieving adequate code quality, verified by passing SonarQube's default quality gate, and for excellent functional correctness, measured by the performance on our hidden test suite. During these 8 weeks, students received daily feedback.

[1] https://github.com/ftsrg-edu/feseip
[2] https://github.com/ftsrg-softeng/openmetromaps
[3] https://www.openmetromaps.org/
[4] https://gtfs.org/

Listing 1. An excerpt from one of the variants' specifications.

A1 Merge Stops:

1. The user selects exactly two stops. The order of selection matters. The stop first selected is called the *primary stop*, and the stop selected second is the *secondary stop*.
2. As a result of the merge, the *primary stop* is modified, and the *secondary stop* is removed from the model.
3. All lines that have a stop at the *secondary stop* will stop at the *primary stop* after the merge.
4. If a line stops at both the *primary* and *secondary stops*:
   (a) If the two selected stops are adjacent (with no other stop in between), the segment defined by the two adjacent stops is removed from the line.
   (b) If the two selected stops are not adjacent (with at least one stop in between), the merge operation is not executed on any line.
   (c) There must be at least two stops on each line. If this is not ensured as a result of the merge operation, the operation will not be executed.

Example Assignment. Students receive a GitHub issue containing the requirements for their variant, with pre-itemized requirements for better clarity. Listing 1 provides an excerpt from such a specification. Typically, students are expected to write 150-200 lines of code. Listing 1 contains one out of five operations from that variant, requiring 40-60 lines of code. The students' code must adhere to a specific interface to enable automatic evaluation. The primary challenge is not coding per se, but analyzing the 25,000 lines of OpenMetroMaps code and comprehending its underlying data structure, before being able to contribute the 200 lines this new feature requires.

In the test design phase, students apply specification-based techniques to create tests. They utilize the arrange, act, assert structure, where standard GTFS files represent the arrange and assert components detailing a map's state pre- and post-operation, and a text file describes the operation. Utilizing GTFS files enables students to practice with established standards unfamiliar to them.

3. Evaluation

In 2024, the assignment was launched within an undergraduate course, engaging around 700 students in Hungarian, English, and German languages. The task commenced on October 8$^{\text{th}}$, with optional phase deadlines: onboarding by October 15$^{\text{th}}$, implementation by October 29$^{\text{th}}$, test design by November 12$^{\text{th}}$, and review by November 26$^{\text{th}}$. The final submission deadline was November 28$^{\text{th}}$, with late submissions[5] accepted until December 5$^{\text{th}}$.

Initially, we posed two research questions:

[5]University policies govern the length of this period, and require a fee to be paid for late submissions.

- *RQ1*: What is the connection between a student's willingness to adhere to the schedule, the quality, and the functional correctness of the code they write?
- *RQ2*: Is identifying students facing difficulties early on possible?

We built a dataset from each student's activity during the semester to answer our research questions. We collected:

- Whether the student passed the home assignment or not;
- Dates of completion for onboarding, implementing, test design, and review phases;
- For onboarding commits: commit date;
- For implementing and review commits: commit date, performance on the hidden test suite, SonarQube quality metrics (SQALE index, numbers/types of bugs, code smells, written unit tests, and their coverage);
- For test design commits: commit date, test suite size, suite correctness (tests passing on the reference implementation/total tests), and suite completeness (mutants detected by suite/total mutants; incorrect tests excluded).

We utilized exploratory data analysis (EDA) methods to analyze patterns in the data, enabling us to examine correlations between student activity timelines, code quality metrics, and functional correctness. We also aimed to identify behavioral indicators of potential challenges during the assignment.

Initially, we excluded all student records involving plagiarism during the home assignment. This resulted in a dataset of 641 students for further analysis (refer to Figure 3). Among these, 69% completed the onboarding phase on time, 29% were late, and 2% gave up before completion. Conversely, just 29% completed the implementing phase on time, with 60% late and 11% dropping out by this point. A trend was observed where those on time in onboarding often fell behind in implementing, whereas few who started late managed to catch up. In the test design phase, 37% were on time, 49% late, and 14% gave up before reaching it. Those delayed in implementation usually continued the pattern into the test design phase, although a few students caught up with the deadlines here. Notably, by the

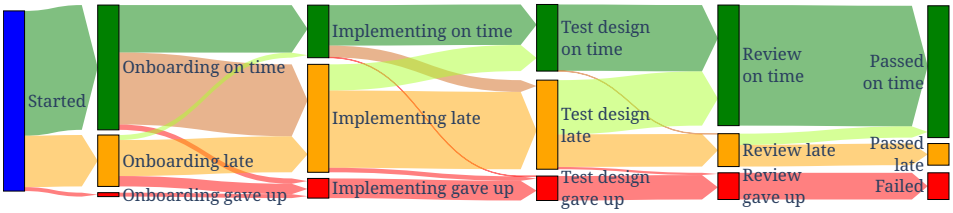
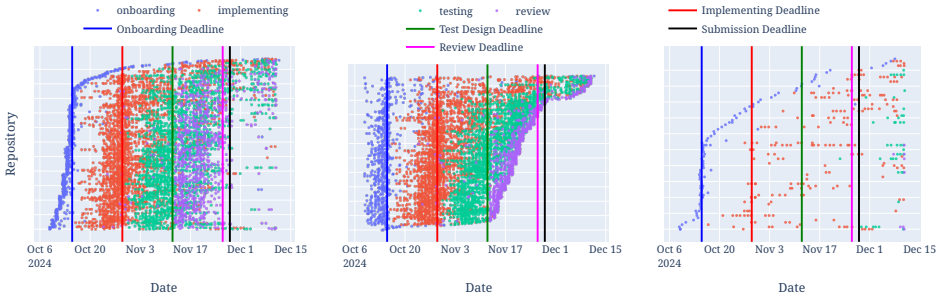


Figure 3. Sankey diagram visualizing the proportion of students completing the home assignment phases on time.

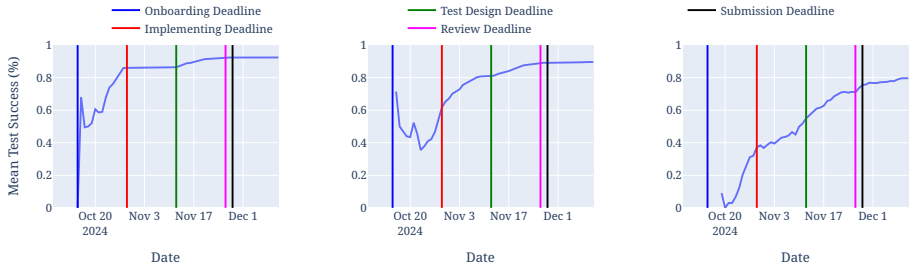
review deadline, 67% were on time, split evenly between those on time and those late in earlier phases, while 18% were late, and 15% gave up. In the two days from the optional review deadline to the final cutoff, an additional 6% completed their review phase and, consequently, the home assignment.

Investigating student progression in further depth (Figure 4), it becomes evident that although a single week sufficed for the onboarding phase, the implementation and test design phases demanded two weeks of student engagement. However, the majority initiated the implementation phase a mere week before its deadline (Figure 4a). Students who commenced the home assignment well ahead of the optional onboarding phase deadline largely managed to avoid late submissions (Figure 4a), whereas late submitters frequently did not commence the onboarding and implementation phases on schedule (Figure 4b). Additionally, Figure 4c shows that nearly all failing students only began the test design phase during the late submission timeframe.



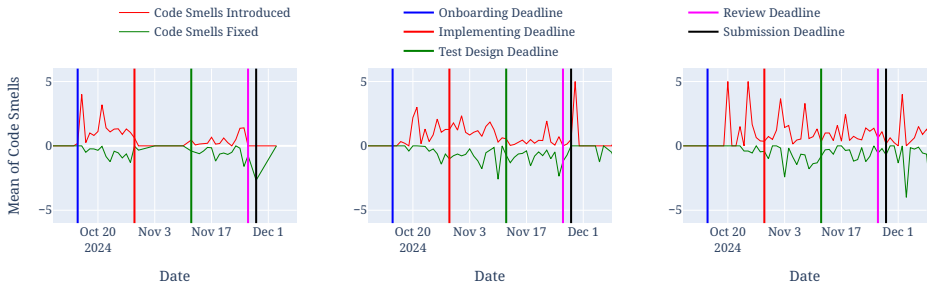
(a) ...in the order of the first commit. **(b)** ...in the order of the last commit. **(c)** ...who failed.

Figure 4. Scatterplot where each point represents a commit of a student's repository...



(a) ...by the implementation deadline. **(b)** ...by the test design deadline. **(c)** ...late.

Figure 5. The mean test success percentage on each day, faceted by the completion of the coding phase...



(a) ...by the implementation deadline.

(b) ...by the test design deadline.

(c) ...late.

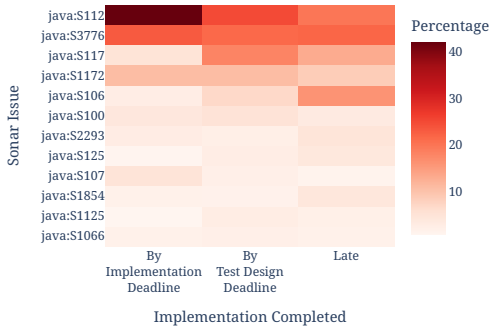
Figure 6. The mean number of code smells introduced (red, above zero) and fixed (green, below zero) on each day, faceted by the completion of the coding phase...

Regarding functional correctness, Figure 5 illustrates students' mean test success rate. Those who completed the implementation by its deadline (Figure 5a) attained an 86% success rate, ultimately rising to 93% after addressing issues revealed by their peers' test suites during the review phase. Conversely (Figure 5b), students who completed the implementation only by the test design deadline, being at most 2 weeks late, achieved 80% by that time and improved to 90% by the assignment's conclusion. Thus, students with delayed completion demonstrated worse functional correctness, requiring more revisions during review. Students who missed even the test design deadline (Figure 5c) reached only 80% by the assignment's end.

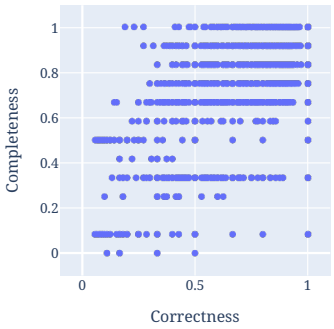
Figure 6 depicts the average change in code quality among students. It reveals that students who met the implementation deadline frequently generated issues during this phase, which they subsequently addressed during the review stage (see Figure 6a). Those completing the implementation by the test design deadline (delayed by up to two weeks) introduced issues over a prolonged period but actively resolved them (refer to Figure 6b) later. Students delayed by more than two weeks not only prolonged issue introduction but also created more issues on average and resolved fewer than the aforementioned groups (Figure 6c).

Analyzing the code quality further, Figure 7a illustrates the most common code smells observed at the end of the project. Table 1 reveals that the second most frequent issue was the high cognitive complexity of student code, indicating poor structural organization. The primary issue among punctual students involved throwing generic exceptions. Notably, students who started late threw fewer exceptions, suggesting a potential neglect in addressing edge cases. Further distinctions include delayed students' non-compliance with naming conventions and a tendency to retain debugging logs in final submissions.

An examination of individual outcomes corroborates prior results. Figure 8 compares the students' project code quality, assessed with SonarQube's SQALE



(a) The heatmap of the frequency of the most common code smells found in the students' code.



(b) The correctness of the students' designed test suite depicted against the completeness of the test suite

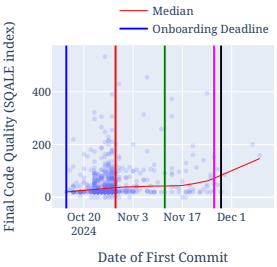
Figure 7

Table 1. The description, severity, and category of the most common code smells found in the student's code.

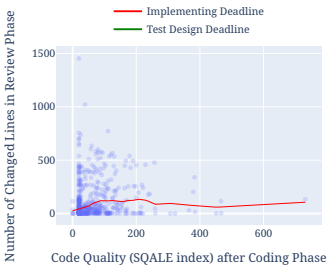
Rule	Description	Severity	Category
java:S112	Generic exceptions should never be thrown	Medium	Intentionality
java:S3776	Cognitive complexity too high	High	Adaptability
java:S117	Variable naming convention not complied with	Low	Consistency
java:S1172	Unused parameters	Medium	Intentionality
java:S106	Logging to standard output	Medium	Adaptability
java:S100	Method naming convention not complied with	Low	Consistency
java:S2293	Diamond operator not used	Low	Intentionality
java:S125	Commented out block of code	High	Intentionality
java:S107	Method has too many parameters	High	Adaptability
java:S1854	Unused assignments	High	Intentionality
java:S1125	Redundant boolean literals	Low	Consistency
java:S1066	Mergeable if statement	High	Intentionality

index, against other measures. Figure 8a illustrates a pattern where projects initiated later show poorer final code quality, particularly those starting near deadlines, implying students sacrificed quality for adherence to the timeline, and chose not to address code quality later. Figure 8b reveals that a higher SQALE index at the end of the implementation indicates a higher number of lines changed in the review phase, suggesting higher challenges in resolving code issues post-implementation. While the SQALE index goes from 0 to 200, the median line changes increase from 20 to 120. Insufficient amount of data exist for higher SQALE indices. Additionally, Figure 8c correlates lower code quality with reduced test success percentage, indicating a decline in functional correctness.

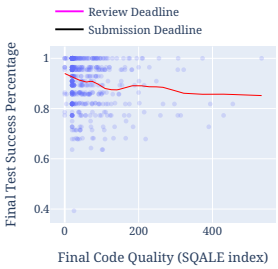
Finally, Figure 7b presents the two evaluation criteria for the student-designed test suites. Although there is a noticeable relationship where increased correctness suggests greater completeness and the reverse, the considerable variance indicates that students probably focused on one metric at a time.



(a) When did the students start the assignment, vs. what was their final code quality.



(b) What was the students' code quality at the end of the coding phase, vs. how much did they modify their implementation in the review phase.



(c) What was the students' final code quality compared to how much of the hidden test suite did their implementation pass at the end.

Figure 8. The connection between the students' work schedule, code quality, and functional correctness.

Discussion. Prior results demonstrate a clear connection between students' adherence to the schedule, code quality, and functional correctness (*RQ1*). A delayed start on the assignment often results in diminished code quality and increased review workloads, which adversely affects functional correctness. However, late starters may possess lower programming skills, contributing to these outcomes. Evaluations would benefit from incorporating data from previous programming courses to exclude this option. Conversely, no definitive early indicators of failure are evident early on (*RQ2*). Although failing students frequently delay their start, this is also true of several students who ultimately succeed, necessitating further investigation.

4. AI-assisted outcome estimation

To deepen our analysis of *RQ2*, whether students facing difficulties can be identified early, we applied an Explainable Boosting Machine (EBM) [14], a transparent, interpretable model that balances predictive performance with explainability. Unlike traditional decision tree-based approaches, EBM often achieves higher accuracy while allowing detailed insight into how individual features influence predictions. This makes it particularly valuable in educational settings, where understanding the reasoning behind model decisions is essential. By examining feature contributions and interaction effects, we aim to uncover which early behaviors most indicate eventual failure, enabling more informed, data-driven interventions.

Following data cleansing and preparation, we generated eight distinct datasets. Weekly thresholds aligning with the phase deadlines and midpoints were established (refer to Table 2), and all data beyond these thresholds were filtered out to finalize

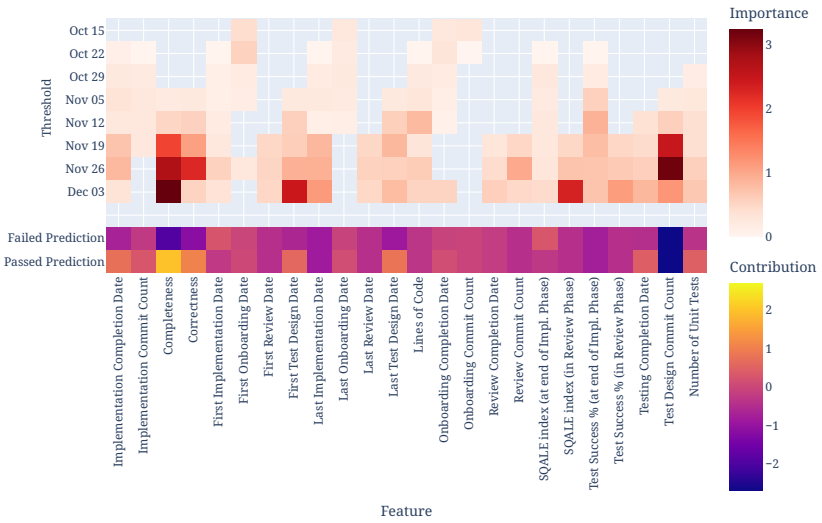


Figure 9. The importance of certain features of the explainable boosting classifier (Importance). The contribution of features to a failing and passing prediction for the sixth dataset (Contribution).

our datasets. Thus, each dataset contained everything known at that threshold. The target variable was whether the student passed the home assignment.

Then, an EBM-based binary classifier was trained for each dataset. Figure 9 presents the training outcomes, whereas Table 2 displays the AUC, sensitivity, and specificity metrics.

The specificity is around 0.5 for the initial two datasets aligned with the onboarding deadline and implementation phase midpoint. This indicates the model failed to identify reasons for student failure, consistently predicting a passing outcome. Referencing Figure 4, merely half of the failing students had commenced work by these points. Those engaged predominantly had not begun the implementation phase. Furthermore, given that numerous students with analogous behavior completed the home assignment successfully, the classifier lacked sufficient data for differentiation.

From the third dataset onward, specificity steadily rises with the addition of more information, while sensitivity slightly diminishes due to a substantial number of new student entries at this stage. For the third and fourth datasets, as the implementation phase concludes and the test design phase reaches its midpoint, specificity surpasses 70%. Key predictors for these datasets include the completion date of the implementation phase, functional correctness, the number of commits, and lines of code authored by students in this phase. Notably, from the fourth dataset, the number of commits in the test design phase and the accuracy and completeness of the test suite become increasingly significant.

Table 2. The AUC, sensitivity, and specificity of the models.

Threshold	AUC	Sens.	Spec.	Threshold	AUC	Sens.	Spec.
October 15	0.740	0.829	0.533	November 12	0.949	0.876	1.000
October 22	0.825	0.924	0.567	November 19	0.967	0.914	1.000
October 29	0.876	0.810	0.700	November 26	0.985	0.971	1.000
November 05	0.929	0.876	0.833	December 03	0.994	0.962	1.000

The fifth (test design deadline) is the first dataset with a specificity of 1.0, demonstrating the model's capability to predict failing students accurately. From this point, sensitivity and AUC rise and remain high, signifying strong predictive performance. Previous trends persist: crucial elements include the commit count during the test design phase and the test suite's correctness and completeness. Additionally, functional correctness, the implementation phase's line count, and the final test design phase commit date gain significance.

Datasets six and seven, corresponding to the review phase's midpoint and deadline, emphasize test suite correctness, completeness, and commit count during test design phase as critical. Notably, the implementation phase's completion date and review phase commit count become more significant than in other datasets. In the seventh dataset, implementation phase features, like lines of code, initial commit date, and SQALE index, become notable. This pattern, shown in Figure 4c, reflects many students who postponed implementation and failed the course.

By the eighth dataset related to late submission, over 75% of students had already completed the home assignment. Key characteristics align with the seventh dataset, but the final commit date in the test design phase, the final SQALE index, implementation phase code lines, and test success percentage (functional correctness) gain more significance.

Using an EBM facilitates the explanation of each feature's contribution to predictions. This is illustrated with a failing and a passing prediction for the sixth dataset (Figure 9, bottom two rows). Negative contributions are linked with failures, while positive ones signify successes. These contributions are largely consistent with the importances in the datasets. For students who fail, this insight aids in giving customized advice aimed at passing the assignment. Conversely, for successful students, it can be used to suggest improvements in features that contributed against that prediction, improving the quality of their solution.

Discussion. Our findings demonstrate the potential for identifying students who are encountering difficulties at an early stage (*RQ2*). However, there is a notable amount of false negative outcomes during the initial month of observation. As depicted in Figure 4, there is an implication that prioritization of support should be directed towards students who show a lack of engagement with their assignments within this initial timeframe. Nevertheless, EBMs continue to serve effectively in the context of prediction interpretation, providing valuable insights into the primary features that contribute to the outcomes. Analyzing these features, we can provide automated recommendations tailored for students who are at risk of failing.

5. Related work

Multiple tools provide interactive and gamified learning experiences tailored for different audiences and skill levels. LearnGitBranching [22] is an example that elucidates the fundamentals of distributed version control. Codio and CodeAcademy cater to professionals, students, and universities [5, 20], covering a wide range of topics. While these platforms offer a simulated environment for learning, they are typically restricted to specific domains, lacking real-world project complexities. CI-based solutions [1, 2, 6, 15], utilizing GitHub Actions, GitLab CI/CD, and Jenkins, are prevalent due to their capacity for continuous, automated feedback, urging students to use industry-standard tools. However, these systems can be limited by the CI platform, presenting challenges in database integration, meeting deadlines, and concealing evaluative aspects from students. Our method aims to integrate the interactive and gamified aspects of simulation tools with the technological stack of CI-based systems.

Extensive research exists on teaching code quality within software engineering education [9, 11]. Gilson et al. [7] examine software quality assessment in a year-long project, emphasizing long-term technical debt management by students. Conversely, this paper assesses code quality in relation to other metrics, such as functional correctness and deadline compliance, over a shorter 7-week timeframe. Meanwhile, Senger et al. [19] investigate the links between assignment solution time, final outcomes, and code quality in small tasks. A notable distinction is that Senger et al. do not provide static analysis results to students and do not aim to teach static analysis.

Most recent research investigating the use of static analysis tools in software engineering education, such as [12, 13, 18], focuses on the security vulnerability detection. In contrast, our paper exclusively focuses on maintainability (SQALE index) and functional correctness.

6. Conclusion

In summary, our assignment framework demonstrates the feasibility and effectiveness of incorporating real-world software engineering practices into educational settings at scale. Utilizing industry-standard tools, best practices, and automated feedback, we offer a learning experience that goes beyond traditional programming tasks, enhancing student participation and readiness for professional software engineering. Our evaluation indicates a correlation between students' willingness to meet optional deadlines and the quality and functionality of their code, and shows that AI tools can identify struggling students during the assignment period. In the future, we plan to use AI-assisted outcome predictions to provide tailored advice for students who are falling behind.

References

[1] X. Bai, M. Li, D. Pei, S. Li, D. Ye: *Continuous delivery of personalized assessment and feedback in agile software engineering projects*, in: Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training, ICSE-SEET '18, New York, NY, USA: Association for Computing Machinery, 2018, pp. 58–67, ISBN: 9781450356602, DOI: 10.1145/3183377.3183387.

[2] J. Bennedsen, T. Böjttjer, D. Tola: *Using GitHub classroom in teaching programming*, in: Proceedings of the 18th International CDIO Conference, 2022, p. 690.

[3] S. Bhalerao, D. Puntambekar, M. Ingle: *Generalizing Agile software development life cycle*, International journal on computer science and engineering 1.3 (2009), pp. 222–226, ISSN: 0975–3397.

[4] J. C. Cortés Ríos, S. M. Embury, S. Eraslan: *A unifying framework for the systematic analysis of Git workflows*, Information and Software Technology 145 (2022), p. 106811, ISSN: 0950-5849, DOI: 10.1016/j.infsof.2021.106811.

[5] D. Croft, M. England: *Computing with Codio at Coventry University: Online virtual Linux boxes and automated formative feedback*, in: Proceedings of the 3rd Conference on Computing Education Practice, CEP '19, New York, NY, USA: Association for Computing Machinery, 2019, ISBN: 9781450366311, DOI: 10.1145/3294016.3294018.

[6] B. P. Eddy, N. Wilde, N. A. Cooper, B. Mishra, V. S. Gamboa, K. M. Shah, A. M. Deleon, N. A. Shields: *A Pilot Study on Introducing Continuous Integration and Delivery into Undergraduate Software Engineering Courses*, in: 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T), 2017, pp. 47–56, DOI: 10.1109/CSEET.2017.18.

[7] F. Gilson, M. Morales-Trujillo, M. Mathews: *How junior developers deal with their technical debt?*, in: Proceedings of the 3rd International Conference on Technical Debt, TechDebt '20, Seoul, Republic of Korea: Association for Computing Machinery, 2020, pp. 51–61, ISBN: 9781450379601, DOI: 10.1145/3387906.3388624.

[8] Y. Jia, M. Harman: *An Analysis and Survey of the Development of Mutation Testing*, IEEE Transactions on Software Engineering 37.5 (2011), pp. 649–678, DOI: 10.1109/TSE.2010.62.

[9] H. Keuning, J. Jeuring, B. Heeren: *A Systematic Mapping Study of Code Quality in Education*, in: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2023, Turku, Finland: Association for Computing Machinery, 2023, pp. 5–11, ISBN: 9798400701382, DOI: 10.1145/3587102.3588777.

[10] B. McHugh: *Beyond transparency: open data and the future of civic innovation*, in: Code for America Press San Francisco, 2013, chap. Pioneering open data standards: The GTFS Story, pp. 125–135, ISBN: 978-0615889085.

[11] D. Nikolić, D. Stefanović, M. Nikolić, D. Dakić, M. Stefanović, S. Koprivica: *Uncovering Determinants of Code Quality in Education via Static Code Analysis*, IEEE Access 12 (2024), pp. 168229–168244, DOI: 10.1109/ACCESS.2024.3426299.

[12] S. Nocera, S. Romano, R. Francese, G. Scanniello: *Software engineering education: Results from a training intervention based on SonarCloud when developing web apps*, Journal of Systems and Software 222 (2025), p. 112308, ISSN: 0164-1212, DOI: 10.1016/j.jss.2024.112308, URL: https://www.sciencedirect.com/science/article/pii/S0164121224003522.

[13] S. Nocera, S. Romano, R. Francese, G. Scanniello: *Training for Security: Results from Using a Static Analysis Tool in the Development Pipeline of Web Apps*, in: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, ICSE-SEET '24, Lisbon, Portugal: Association for Computing Machinery, 2024, pp. 253–263, ISBN: 9798400704987, DOI: 10.1145/3639474.3640073.

[14] H. Nori, S. Jenkins, P. Koch, R. Caruana: *InterpretML: A Unified Framework for Machine Learning Interpretability*, ArXiv abs/1909.09223 (2019), URL: https://api.semanticscholar.org/CorpusID:202712518.

[15] M. Ohtsuki, K. Ohta, T. Kakeshita: *Software engineer education support system ALECSS utilizing DevOps tools*, in: Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services, iiWAS '16, New York, NY, USA: Association for Computing Machinery, 2016, pp. 209–213, ISBN: 9781450348072, DOI: 10.1145/3011141.3011200.

[16] S. Ouhbi, N. Pombo: *Software Engineering Education: Challenges and Perspectives*, in: 2020 IEEE Global Engineering Education Conference (EDUCON), 2020, pp. 202–209, DOI: 10.1109/EDUCON45650.2020.9125353.

[17] N. B. Ruparelia: *Software development lifecycle models*, ACM SIGSOFT Software Engineering Notes 35.3 (2010), pp. 8–13, DOI: 10.1145/1764810.1764814.

[18] A. Sanders, G. S. Walia, A. Allen: *Assessing common software vulnerabilities in undergraduate computer science assignments*, Journal of The Colloquium for Information Systems Security Education 11.1 (2024), pp. 8–8, DOI: 10.53735/cisse.v11i1.179.

[19] A. Senger, S. H. Edwards, M. Ellis: *Helping Student Programmers Through Industrial-Strength Static Analysis: A Replication Study*, in: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1, SIGCSE 2022, Providence, RI, USA: Association for Computing Machinery, 2022, pp. 8–14, ISBN: 9781450390705, DOI: 10.1145/3478431.3499310.

[20] J. H. Sharp: *Using Codecademy Interactive Lessons as an Instructional Supplement in a Python Programming Course.* Information Systems Education Journal 17.3 (2019), pp. 20–28.

[21] I. Sommerville: *Software Engineering*, International Computer Science Series, Pearson, 2011, ISBN: 9780137053469.

[22] G. Wagner, L. Thurner: *Teaching Tip: Rethinking How We Teach Git: Pedagogical Recommendations and Practical Strategies for the Information Systems Curriculum*, Journal of Information Systems Education 36.1 (2025), pp. 1–12, DOI: 10.62273/BTKM5634.

[23] T. Winters, T. Manshreck, H. Wright: *Software engineering at Google: Lessons learned from programming over time*, O'Reilly Media, Inc., 2020.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 65–77
DOI: 10.17048/fmfai.2025.65

Hungarian case study on automated detection of body-shaming comments using machine learning

Franciska N. Gőz, Erika B. Varga

Institute of Information Technology,
Faculty of Mechanical Engineering and Informatics
University of Miskolc, H-3515 Miskolc, Hungary
gozfanni@gmail.com
erika.b.varga@uni-miskolc.hu

Abstract. Social media facilitates online interactions but also enables body-shaming comments which are often ambiguous. This paper presents a machine learning-based approach for detecting Hungarian body-shaming comments, an underrepresented area in `NLP`. A dataset of Facebook comments was collected and expanded with synthetic data. Using `HuSpaCy` and `HuBERT`, logistic regression and `MLP` classification models were trained with `TF-IDF` and `SBERT` embeddings. The best-performing model achieved 88 % accuracy, demonstrating the potential of `NLP` techniques for moderating harmful online content in low-resource languages. The results highlight key challenges, including category overlap and class imbalance, emphasizing the need for context-aware classification methods in automated content moderation.

Keywords: Hungarian text analysis, toxic comment filtering, social media moderation, body-shaming detection, machine learning classification

AMS Subject Classification: 68T05, 68T07, 68T50

1. Introduction

Social networking sites offer remarkable opportunities for communication and connection, but at the same time they also serve as fertile ground for the spread of harmful behaviors. Damaging comments, encompassing various forms such as body shaming, hate speech, cyberbullying, and online harassment, have become increasingly widespread [15]. The widespread nature of these comments contributes to a

toxic online environment, affecting not only individual well-being but also shaping social attitudes and potentially fueling real-world discriminatory behaviors [16].

The exponential growth of user-generated content has facilitated the spread of such harmful language, posing serious problems in maintaining a respectful online environment [2]. The automatic detection and moderation of these harmful comments presents significant challenges due to the large amount of online content, the diversity of language, and the difficulty in distinguishing harmful intent from protected free speech [7].

Body shaming is defined in [25] as a type of negative social interaction which involves derogatory comments about an individual's physical appearance, often leading to diminished self-esteem, social withdrawal, and even mental health issues such as depression and eating disorders [11]. Social media platforms have attempted to address the spread of body-shaming content through community guidelines and moderation systems. However, these efforts are often insufficient due to the great volume of content and the ambiguous nature of body-shaming remarks, which can be disguised as humor [8]. The urgency to address this issue stems not only from its psychological impact but also from the legal and ethical obligations of these platforms to provide a safe environment for their users [1].

Effective interventions require a twofold approach. Social networking platforms should enhance their moderation systems with machine learning techniques to automatically detect and classify harmful content. These methods can help scale the identification and removal of body-shaming remarks, even when they are disguised [10]. On the other hand, comprehensive legal frameworks, such as the European Union's Digital Services Act (DSA) [21], should be established to hold platforms accountable for harmful content while balancing the principles of freedom of speech [27].

This study contributes to these efforts by exploring the development of a classification model for detecting body-shaming comments in Hungarian. By integrating machine learning techniques and considering the complexities of both negative and ambiguous remarks, the proposed solution aims to support the automated moderation systems of social media platforms.

The novelty of this work lies primarily in the creation of the first annotated dataset of Hungarian body-shaming comments and in presenting a case study for using Hungarian text processing tools.

2. Related works

Body shaming has been widely recognized as a significant social issue, particularly in the context of social media. Schlüter et al. [24] conducted an exploratory study to provide a scientifically grounded definition and classification of body shaming. They define body shaming as an unrepeated act in which individuals express unsolicited, predominantly negative opinions about another person's body, often perceived negatively by the target. Importantly, their findings highlight the dimensional nature of body shaming, ranging from well-meant advice to overtly malicious

insults. The study emphasizes the distinction between body shaming and related concepts, such as appearance teasing, trolling, and cyberbullying, noting that body shaming is not necessarily repeated nor always intentional. Since body shaming is harmful to the victims' physical and mental health, numerous studies investigated its impacts, for example on women's body image concerns [19], on eating disorders [12, 23, 26], and on mental well-being [4].

Body shaming is increasingly prevalent on social media platforms [6]. While these platforms have their own community guidelines to regulate and remove harmful posts and comments, their policies are shaped by the legal framework of the countries in which they operate. In the United States, for example, the First Amendment of the Constitution guarantees broad freedom of speech, allowing individuals to express their opinions freely, even if they are offensive to others. However, specific cases like threats or defamation may still be restricted. Since major social media platforms like Facebook, Instagram, and Twitter are based in the U.S., they primarily adhere to these legal principles. While the platforms' community guidelines prohibit harassment, hate speech, and body-shaming content, their enforcement is not legally mandated but rather a voluntary application of their policies. According to the community standards of Facebook and Instagram, hate speech and harassment, including content targeting individuals based on their physical appearance are prohibited. Body shaming explicitly falls under this category, and such posts are subject to removal. Twitter's rules also ban behavior that harasses or intimidates others, including body shaming. The platform takes actions to suspend or ban offending users.

The U.S. prioritizes freedom of speech as a fundamental right, emphasizing minimal government intervention in online content moderation. This framework allows platforms significant authority in enforcing their policies without strict government oversight. In contrast, the European Union, through the DSA, imposes stricter obligations on social media platforms to tackle harmful content, including hate speech and harassment. The DSA emphasizes protecting users from harmful online behavior, requiring platforms to remove illegal content promptly, increase transparency in content moderation policies, and offer effective appeal mechanisms for users whose content is removed [3].

In this context, investigations on classifying comments on body shaming by means of machine learning algorithms seems to bring benefits for the users of social networking sites. What makes the task more complicated is to identify a writer's real intent. Body-shaming remarks are often intended as humor from the speaker's perspective or expressed through ambiguous language. This subjectivity makes it difficult, even for humans, to determine whether a particular comment falls into the category of body shaming. Consequently, the automatic detection of such remarks presents a significant challenge [28].

Some recent studies concentrate on detecting and analyzing body-shaming online comments. In [5] sentiment analysis is performed on Twitter comments using Naive Bayes Classifier. The dataset consists of 1,000 training tweets and 329 test tweets. The algorithm achieved an accuracy of 80.55 % and highlighted key words

like "overweight" and "thin". The research underlines the potential of Naive Bayes for effective classification of negative and positive sentiments, though it acknowledges challenges in dealing with context and linguistic issues.

Jaman et al. [14] use the Naive Bayes Classifier to detect body-shaming sentiments in comments on YouTube beauty vlogs. The study involves 33,044 comments, of which 986 were labeled as body-shaming after manual validation. The model achieved a high accuracy of 98.48 %, demonstrating the effectiveness of tailored preprocessing and dataset splits.

Grasso et al. [9] examine the detection and classification of body shaming content in Italian social media applying contextual word embeddings and transformer-based architectures. The study highlights challenges in multilingual settings, such as data scarcity and linguistic diversity, and proposes advanced methods to overcome these.

Reddy et al. [20] explore body shaming online content detection in low-resource languages, using transfer learning and multilingual embeddings to bridge the gap caused by limited datasets. The authors show that pre-trained language models like `mBERT` can improve classification performance when fine-tuned on small, annotated corpora.

In our experiments, we collected comments from Facebook and supplemented them with synthetic texts to develop and compare machine learning models for classifying comments about an individual's physical appearance into six categories. Five of these classes fall within the domain of body shaming, such as negative opinions on overweight, underweight, height, skin tone and body hair; while the sixth class is to contain sentiments on physical traits that are not considered as body shaming. Our experiments focus on Hungarian, an underrepresented language in social media. To the best of our knowledge, this is the first scientific paper that addresses the filtering of Hungarian-language texts on this specific topic.

3. Methods and data

3.1. Hungarian text processing tools

For preprocessing we used `HuSpaCy` [18], a Hungarian adaptation of the `spaCy` library providing tokenization, lemmatization, part-of-speech tagging, dependency parsing, and named entity recognition. `HuSpaCy` has been optimized for Hungarian linguistic resources and offers efficient pipelines suitable for machine learning applications.

For embeddings we employed `HuBERT` [17], a transformer-based language model following the `BERT` architecture [13] and trained on `Webkorpusz` 2.0, a large Hungarian corpus. To obtain sentence-level representations, we applied the `Sentence-BERT` (`SBERT`) approach [22] on top of `HuBERT` outputs. This combination yields rich contextual embeddings that have proven effective in Hungarian `NLP` tasks.

These tools ensured reliable preprocessing and rich representations for classification while keeping the methodology comparable with prior work in low-resource

language settings.

3.2. Data collection and synthetic data generation

The initial dataset was translated from comments collected from Facebook. The focus was on comments related to body-shaming, which were identified and annotated manually by one of the authors, based on her interpretation of the content. These comments represented diverse categories of harmful speech aimed at various physical traits, such as weight, height, body hair, and skin tone. Since annotation was carried out by a single annotator, we acknowledge that subjectivity may affect the labeling. In particular, ambiguous cases sometimes arise, for example when a comment such as *"Hűha, jó sokat fogytál!" – "Wow, you've lost a lot of weight!"* could be interpreted either as a positive remark (non-toxic) or as a negative body-related comment (skinny). The annotated comments were finally organized into five predefined categories:

- Body hair
  (e.g., *"Úgy néz ki a lábad, mintha évek óta nem találkozott volna borotvával." – "Your legs look like they haven't seen a razor in years."*)
- Height
  (e.g., *"Gondolom, nálad nem opció a legfelső polcra pakolás." – "I bet you can't even reach the top shelf."*)
- Fat (e.g., *"Toka az egész fejed." – "Your double chin is your entire face."*)
- Skinny (e.g., *"Lapos vagy, mint egy deszka." – "You're flat as a board."*)
- Skin tone
  (e.g., *"Annyira világos a bőröd, hogy egy nyaralás után is alig látszik változás." – "Your skin is so pale, even after a vacation, there's no difference."*)

To enrich the dataset and increase its diversity, synthetic comments were generated using ChatGPT-4. In the prompts, we specified the target category (e.g., body hair, skin tone), instructed the model to generate a short (1-2 sentences) negative comment in an informal style resembling Facebook posts, and included examples taken from our manually annotated data. Approximately 5 % of the final dataset consists of such synthetic comments. We did not evaluate model performance separately on real and synthetic comments due to the small dataset size, but the synthetic data was only used to increase diversity and balance across categories, not to replace authentic user comments.

The dataset was further augmented with synthetic statements that express positive sentiments regarding a person's physical appearance. These comments were labeled as the "non-toxic" category, e.g. *"Nagyon szép a bőröd ezen a fotón, természetes ragyogás!" – "Your skin looks beautiful in this photo, with a natural glow!"*. In this way, we can investigate whether positive sentiments can be clearly distinguished from negative ones through classification.

3.3. Dataset statistics

The dataset comprises 330 labeled comments with approximately equal representation across the six categories. On average, there are 55 comments per category, amounting to a balanced dataset for training purposes. The detailed statistics are presented in Tables 1 and 2.

Table 1. Dataset statistics before tokenization.

Label	Number of comments	Num. of words				Num. of chars			
		Avg	Std	Min	Max	Avg	Std	Min	Max
bodyhair	63	11.25	2.61	5	18	69.86	13.43	32	102
height	63	10.13	4.76	5	25	65.17	29.25	26	144
fat	55	8.16	3.41	2	16	48.87	23.02	11	102
skinny	64	8.75	2.36	2	16	52.28	15.8	12	110
skintone	35	10.14	1.83	7	14	61.97	10.52	42	85
non-toxic	50	8.92	1.12	6	11	56.16	6.22	47	70
Total	**330**								
Average	**55**	9.56	2.68	4.5	16.67	59.05	16.37	28.33	102.17

Table 2. Dataset statistics after tokenization.

Label	Number of comments	Num. of words				Num. of chars			
		Avg	Std	Min	Max	Avg	Std	Min	Max
bodyhair	63	5.43	1.35	2	9	37.01	8.48	17	62
height	63	5.05	2.05	2	11	33.48	14.81	14	86
fat	55	3.98	1.62	1	8	25.82	12.48	4	59
skinny	64	4.27	1.54	1	10	26.69	10.41	8	65
skintone	35	4.94	1.45	3	8	31	9.09	15	51
non-toxic	50	4.72	0.97	3	7	34.64	8.11	17	59
Total	**330**								
Average	**55**	4.73	1.5	2	8.83	31.44	10.57	12.5	63.67
Total	**330**								
Average	**55**	9.56	2.68	4.5	16.67	59.05	16.37	28.33	102.17

4. Results

The aim of this research is to build a model that can classify social media comments into predefined categories. The first step was to prepare the collected data before text processing. This involved tokenizing and lemmatizing words, and then tranforming sentences into numerical vectors using either `TF-IDF` vectorization or `SBERT` embeddings. Next, the data set was randomly split into training and test sets, and the training set was balanced by `SMOTE` technique (Synthetic Minority Oversampling Technique). Two models were considered for the special purpose: a simple logistic regression model and an MLP neural network. The neural network model was configured with 3 hidden layers containing 100, 50, and 25 neurons respectively; applies ReLU activation function and was optimized using the Adam

algorithm with a maximum of 300 iterations. Finally, four different setups were investigated:

1. Logistic regression model (sentences prepared with `TF-IDF` vectorization)
2. Logistic regression model using `SBERT`
3. `MLP` model (sentences prepared with `TF-IDF` vectorization)
4. `MLP` model using `SBERT`

We applied 5-fold cross-validation for all four model configurations in order to compensate for the randomness of a single train-test split and to ensure more generalizable performance estimates. In each configuration we report the best-performing results (highest accuracy) obtained across the folds.

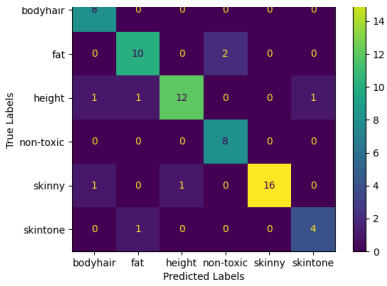
4.1. Logistic Regression model

Logistic regression is a widely used model for text classification tasks due to its simplicity and interpretability. It is particularly effective for categorizing text data into multiple predefined classes, even when only a limited amount of data is available. This makes it a useful algorithm for handling smaller datasets.

Logistic regression

	precision	recall	f1-score	support
bodyhair	0.80	1.00	0.89	8
fat	0.83	0.83	0.83	12
height	0.92	0.80	0.86	15
non-toxic	0.80	1.00	0.89	8
skinny	1.00	0.89	0.94	18
skintone	0.80	0.80	0.80	5
accuracy			0.88	66
macro avg	0.86	0.89	0.87	66
weighted avg	0.89	0.88	0.88	66

(a) Evaluation.



(b) Confusion matrix.

Figure 1. Results for the Logistic regression model with `TF-IDF`.

As depicted in Figure 1a and 1b the model achieved an overall accuracy of 88 %, correctly classifying 88 % of all examples. The macro average represents the average performance across all classes (precision: 0.86, recall: 0.89, F1-score: 0.87), balancing the influence of smaller and larger classes. The weighted performance average, which considers class support, yielded approximately 0.88 for all key metrics. Notably, despite their low support, the "bodyhair" and "non-toxic" categories achieved 100 % recall, meaning they were identified without errors. On the other hand, the "skinny" class with the highest support was predicted with 100 % precision, meaning that no other categories were misclassified here. The model produced the worst performance in the case of the "skintone" class, which has the lowest support in the test set and the lowest representation in the whole dataset.

4.2. Logistic Regression model using SBERT

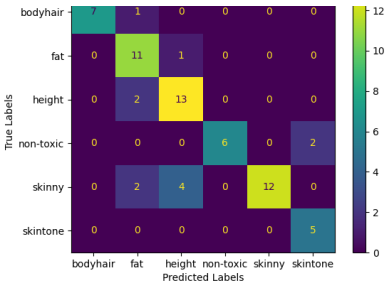
The goal of SBERT is to accurately measure the semantic similarity between sentences. It generates sentence-level embeddings, which provide a numerical representation of sentence semantics, thereby enabling the evaluation of multi-sentence sentiments. The results of the logistic regression model using SBERT embeddings are summarized in Figure 2a and 2b.

The model achieved an accuracy of 82 %, lower than that of traditional logistic regression. The reason for this is that the dataset contains mainly one-sentence long comments and the SBERT technique does not have significant effect here. Nevertheless, the model produced 100 % precision for the class with the highest support ("skinny") and also for the classes having low support ("bodyhair" and "non-toxic"). It is notable, that the model could produce 100 % recall in the case of the "skintone" category, which has the lowest support in the test set and the lowest representation in the whole dataset.

SBERT - Logistic regression

	precision	recall	f1-score	support
bodyhair	1.00	0.88	0.93	8
fat	0.69	0.92	0.79	12
height	0.72	0.87	0.79	15
non-toxic	1.00	0.75	0.86	8
skinny	1.00	0.67	0.80	18
skintone	0.71	1.00	0.83	5
accuracy			0.82	66
macro avg	0.85	0.85	0.83	66
weighted avg	0.86	0.82	0.82	66

(a) Evaluation.



(b) Confusion matrix.

Figure 2. Results for the Logistic Regression model with SBERT embeddings.

4.3. MLP model

The Multi-Layer Perceptron (MLP) is a neural network model adaptable to various problems through optimization of the number of hidden layers, neurons, and learning parameters. This flexibility allows it to perform effectively on text classification tasks. For our data set, the model achieved an accuracy of 85 %, which is a bit lower than the performance of logistic regression. The lower performance may be due to the small dataset size. The detailed evaluation is displayed in Figure 3a and 3b.

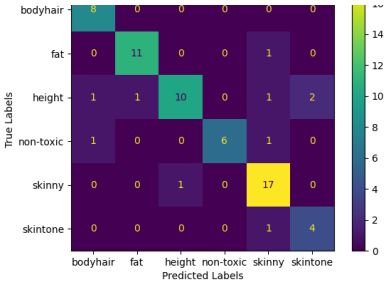
In comparison with logistic regression, this model could not predict the class with the highest support ("skinny") with 100 % precision or recall. On the other hand, it produced 100 % precision in the case of the "non-toxic" class and 100 % recall in the case of the "bodyhair" class, though these have low support in the test set. What is common with logistic regression, the MLP model produced the worst

performance in the case of the "skintone" class, which has the lowest support in the test set and the lowest representation in the whole dataset.

MLP

	precision	recall	f1-score	support
bodyhair	0.80	1.00	0.89	8
fat	0.92	0.92	0.92	12
height	0.91	0.67	0.77	15
non-toxic	1.00	0.75	0.86	8
skinny	0.81	0.94	0.87	18
skintone	0.67	0.80	0.73	5
accuracy			0.85	66
macro avg	0.85	0.85	0.84	66
weighted avg	0.86	0.85	0.85	66

(a) Evaluation.



(b) Confusion matrix.

Figure 3. Results for the `MLP` model with `TF-IDF`.

4.4. `MLP` model using `SBERT`

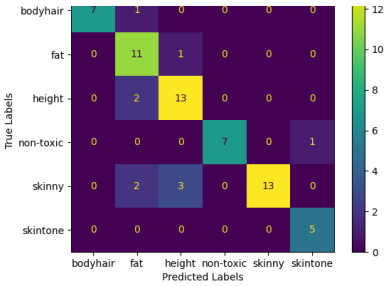
Figure 4a and 4b summarize the results of the `MLP` model using `SBERT` sentence embeddings. The model achieved an overall accuracy of 85 %, which is the same as in the case of the traditional `MLP` model.

The use of `SBERT` yields some similar results than in the case of the logistic regression model. Namely, the model produced 100 % precision for the class with the highest support ("skinny") and also for the classes having low support ("bodyhair" and "non-toxic"). Also, the model could produce 100 % recall in the case of the "skintone" category, which has the lowest support in the test set and the lowest representation in the whole dataset. `SBERT` performs more effectively with `MLP`, as its F1-scores match or exceed those of logistic regression across all categories.

SBERT - MLP

	precision	recall	f1-score	support
bodyhair	1.00	0.88	0.93	8
fat	0.69	0.92	0.79	12
height	0.76	0.87	0.81	15
non-toxic	1.00	0.88	0.93	8
skinny	1.00	0.72	0.84	18
skintone	0.83	1.00	0.91	5
accuracy			0.85	66
macro avg	0.88	0.88	0.87	66
weighted avg	0.88	0.85	0.85	66

(a) Evaluation.



(b) Confusion matrix.

Figure 4. Results for the `MLP` model with `SBERT` embeddings.

To provide a more transparent overview of the model performance across classes, Table 3 summarizes accuracy, macro-F1 scores and the support of each class in the

test set. This highlights the impact of small-sample categories on model performance.

Table 3. Summary of evaluation metrics across models.

Model	Accuracy	Macro-F1	Support (per class)
LogReg (`TF-IDF`)	0.88	0.87	[8, 12, 15, 8, 18, 5]
LogReg (`SBERT`)	0.82	0.83	[8, 12, 15, 8, 18, 5]
MLP (`TF-IDF`)	0.85	0.84	[8, 12, 15, 8, 18, 5]
MLP (`SBERT`)	0.85	0.87	[8, 12, 15, 8, 18, 5]
Ensemble model	**0.83**	**0.86**	**66** (incl. all classes)

5. Conclusions

Body shaming on social media platforms is a significant social and technological challenge. Despite existing moderation efforts, the high volume of content and the complexity of body-shaming remarks make automated detection difficult. This study demonstrates the potential of machine learning techniques in classifying body-shaming comments, particularly in the Hungarian language, which is underrepresented in such research.

The research findings demonstrate that `HuBERT` and `HuSpaCy` are highly effective tools for analyzing Hungarian texts. Their integration enabled the development of efficient classification models, even when working with a small corpus. Logistic regression models offered simplicity and interpretability, achieving an accuracy of up to 88 %. At the same time, `MLP` models utilizing `SBERT` embeddings provided enhanced flexibility in handling ambiguous or complex linguistic context. The models performed particularly well in distinguishing harmless content (the "non-toxic" class) and identifying body-shaming related to body hair, though challenges remained with underrepresented categories like "skin tone".

These findings emphasize the need for tailored machine learning techniques in social media moderation. Future research should focus on addressing class imbalance, expanding datasets with diverse linguistic and cultural contexts, and exploring advanced deep learning architectures to further improve classification accuracy.

6. Limitations and future work

The main limitations of this study are the small dataset size, the underrepresentation of certain categories, and the inherent overlap between them (e.g., skinny vs. fat). These issues are reflected in typical errors, such as misclassifying "Your skin is so pale, even after a vacation there's no difference." as non-toxic instead of skin tone, or labeling "You're flat as a board." as fat instead of skinny. Another limitation is that annotation was carried out by a single annotator, which may introduce

subjectivity, and that comments were analyzed in isolation without conversational context, potentially obscuring pragmatic signals (e.g., irony, humor). Due to limited dataset size and computational constraints, we did not include multilingual transformer baselines such as `mBERT` or `XLM-R` in the current study. Future work should focus on enlarging the dataset, improving class balance, and applying more context-aware models to better handle ambiguous or ironic expressions.

7. Ethical and legal compliance

All data were collected from publicly available Facebook comments. Personally identifiable information was removed during preprocessing to ensure full anonymity. The released dataset contains only anonymized text and complies with the ethical standards for research on social media content as well as the European Union's GDPR regulations.

8. Code and data availability

The code used for training and evaluation (requires `Python` 3.10.0, `spacy` 3.7.4, `huspacy` 0.12.1, and `sentence_transformers` 3.2.1) as well as the anonymized dataset of Hungarian body-shaming comments are openly available at `https://github.com/Fanni98/Diplomaterv`.

References

[1] G. Aitchison, S. Meckled-Garcia: *Against Online Public Shaming: Ethical Problems with Mass Social Media*, Social Theory and Practice 47.1 (2021), pp. 1–31, URL: https://www.jstor.org/stable/45378050.

[2] A. Balayn, J. Yang, Z. Szlávik, A. Bozzon: *Automatic Identification of Harmful, Aggressive, Abusive, and Offensive Language on the Web: A Survey of Technical Biases Informed by Psychology Literature*, ACM Transactions on Social Computing 4.3 (2021), pp. 1–56, DOI: 10.1145/3479158.

[3] A. Candeub: *The Digital Services Act, the First Amendment, and deputised surveillance*, Journal of Media Law 16.1 (2024), pp. 65–73, DOI: 10.1080/17577632.2024.2362479.

[4] F. Deviantony, Y. Fitria, R. Rondhianto, N. Pramesuari: *An in depth review of body shaming phenomenon among adolescent: Trigger factors, psychological impact and prevention efforts*, South African Journal of Psychiatry 30 (2024), p. 2341, DOI: 10.4102/sajpsychiatry.v30i0.2341.

[5] K. Diantoro, Rinaldo, A. Sitorus, A. Rohman: *Analyzing the Impact of Body Shaming on Twitter: A Study Using Naive Bayes Classifier and Machine Learning*, Digitus: Journal of Computer Science Applications 1.1 (2023), pp. 11–25, DOI: 10.61978/digitus.v1i1.58.

[6] G. Fioravanti, S. B. Benucci, V. Vinciarelli, S. Casale: *Body shame and problematic social networking sites use: the mediating effect of perfectionistic self-presentation style and body image control in photos*, Curr Psychol 43 (2024), pp. 4073–4084, DOI: 10.1007/s12144-023-04644-8.

[7] C. Gehweiler, O. Lobachev: *Classification of intent in moderating online discussions: An empirical evaluation*, Decision Analytics Journal 10 (2024), doi: 10.1016/j.dajour.2024.100418.

[8] V. Gongane, M. Munot, A. Anuse: *Detection and moderation of detrimental content on social media platforms: current status and future directions*, Social Network Analysis and Mining 12.1 (2022), p. 129, doi: 10.1007/s13278-022-00951-3.

[9] F. Grasso, A. Valese, M. Micheli: *Body-Shaming Detection and Classification in Italian Social Media*, in: In Natural Language Processing and Information Systems, 2024, doi: 10.1007/978-3-031-70239-6_18.

[10] H. Han, E. A. M. Asif, N. Sarhan, Y. Ghadi, B. Xu: *Innovative deep learning techniques for monitoring aggressive behavior in social media posts*, Journal of Cloud Computing 13.19 (2024), doi: 10.1186/s13677-023-00577-6.

[11] G. Holland, M. Tiggemann: *A systematic review of the impact of the use of social networking sites on body image and disordered eating outcomes*, Body Image 17 (2016), pp. 100–110, doi: 10.1016/j.bodyim.2016.02.008.

[12] A. Hummel, A. Smith: *Ask and you shall receive: Desire and receipt of feedback via Facebook predicts disordered eating concerns*, International Journal of Eating Disorders 48.4 (2015), pp. 436–442, doi: 10.1002/eat.22336.

[13] D. Jacob, M. Chang, K. Lee, K. Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, North American Chapter of the Association for Computational Linguistics, 2019.

[14] J. Jaman, H. Hannie, M. Simatupang: *Sentiment Analysis of the Body-Shaming Beauty Vlog Comments*, in: In Proceedings of the 7th Mathematics, Science, and Computer Science Education International Seminar, 2019, doi: 10.4108/eai.12-10-2019.2296530.

[15] E. Jane: *Online Abuse and Harassment*, in: The International Encyclopedia of Gender, Media, and Communication, 2020, doi: 10.1002/9781119429128.iegmc080.

[16] M. Merino, J. Tornero-Aguilera, A. Rubio-Zarapuz, C. V. Tobaldo, A. Martín-Rodríguez, V. Clemente-Suárez: *Body Perceptions and Psychological Well-Being: A Review of the Impact of Social Media and Physical Measurements on Self-Esteem and Mental Health with a Focus on Body Image Satisfaction and Its Relationship with Cultural and Gender Factors*, Healthcare, doi: 10.3390/healthcare12141396.

[17] D. Nemeskey: *Natural Language Processing methods for Language Modeling*, PhD thesis, Eötvös Loránd University, 2020.

[18] G. Orosz, G. Szabó, P. Berkecz, Z. Szántó, R. Farkas: *Advancing Hungarian Text Processing with HuSpaCy Efficient and Accurate NLP Pipelines*, Speech, and Dialogue 14102 (2023), doi: 10.1007/978-3-031-40498-6_6.

[19] A. Raspovic, I. Prichard, A. Salim, Z. Yager, L. Hart: *Body image profiles combining body shame, body appreciation and body mass index differentiate dietary restraint and exercise amount in women*, Body Image 46 (2023), pp. 117–122, doi: 10.1016/j.bodyim.2023.05.007.

[20] V. Reddy, H. Abburi, N. Chhaya, T. Mitrovska, V. Varma: *You Are Big, S/he Is Small Detecting Body Shaming in Online User Content*, in: In: Social Informatics, 2022, doi: 10.1007/978-3-031-19097-1_25.

[21] *Regulation (EU) 2022/2065 of the European Parliament and of the Council on a Single Market For Digital Services and amending Directive 2000/31/EC (Digital Services Act)*, Official Journal of the European Union, 2022.

[22] N. Reimers, I. Gurevych: *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*, in: Conference on Empirical Methods in Natural Language Processing, p. 2019.

[23] S. Santarossa, S. Woodruff: *SocialMedia: Exploring the Relationship of Social Networking Sites on Body Image, Self-Esteem, and Eating Disorders*, Social Media + Society 3 (2017), doi: 10.1177/2056305117704407.

[24] C. Schlüter, G. Kraag, J. Schmidt: *Body Shaming: an Exploratory Study on its Definition and Classification*, International Journal of Bullying Prevention 5 (2021), doi: 10.1007/s42380-021-00109-3.

[25] C. Schlüter, G. Kraag, J. Schmidt: *Body Shaming: an Exploratory Study on its Definition and Classification*, Int. Journal of Bullying Prevention 5 (2023), pp. 26–37, doi: 10.1007/s42380-021-00109-3.

[26] K. Suhag, S. Rauniyar: *Social Media Effects Regarding Eating Disorders and Body Image in Young Adolescents*, Cureus 16.4 (2024), doi: 10.7759/cureus.58674.

[27] A. Turillazzi, M. Taddeo, L. Floridi, F. Casolari: *The digital services act: an analysis of its ethical, legal, and social implications*, Law, Innovation and Technology 15.1 (2023), pp. 83–106, doi: 10.1080/17579961.2023.2184136.

[28] B. Vidgen, A. Harris, D. Nguyen, R. Tromble, S. Hale, H. Margetts: *Challenges and frontiers in abusive content detection*, in: In Proceedings of the Third Workshop on Abusive Language Online, pp. 88–93, url: https://aclanthology.org/W19-3509.pdf.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 78–89
DOI: 10.17048/fmfai.2025.78

LLM-based framework to support the construction of valid formal models

Gábor Guta[a], **Gábor Kusper**[b]

[a]Axonmatics Kft.
gabor.guta@axonmatics.com

[b]Faculty at Eszterházy Károly University,
Mathematics and Informatics Institute,
gkusper@aries.ektf.hu

Abstract. The use of large language models (LLMs) in software development is becoming increasingly widespread, despite well-known concerns regarding their reliability. A significant risk arises from relying on poorly understood approximate solutions that may subtly introduce errors into the final system. A key barrier to the adoption of formal modeling – beyond the steep learning curve of formal specification languages – is the additional abstraction layer, which can be as difficult to maintain as the source code itself. This complexity persists even when the formal specification can generate code directly. Another challenge is that, while tools for verifying properties of formal models are well-established, the initial translation of a mental model into a formal one often results in invalid or imprecise representations.

We propose a tool which facilitates the validation of formal models generated by LLMs from natural language specifications. The validation process involves two steps: first, the formal model is translated back into natural language using a deterministic, easily verifiable rule-based method; second, the author of the original specification validates this reformulated version. This human-in-the-loop method mitigates the risks associated with LLM black-box generation by enabling explicit semantic verification of the model.

Keywords: LLM, formal specification, human-in-the-loop

1. Introduction

Recent advances in large language models (LLMs) have democratized access to code and specification generation. Their natural language interfaces make them

immediately useful to developers, designers, and even non-technical stakeholders. However, their logical consistency and reliability remain problematic, especially when applied in a business-critical environment.

Formal specification and modeling techniques, such as UML, TLA+, and Alloy, provide solid foundations for correctness, verification, and maintainability, but still have very limited adaptation. This is probably due to various challenges like:

- adoption is hindered by steep learning curves for language syntax and semantics;
- maintaining a separate formal abstraction layer alongside code can be costly and error prone;
- LLM-generated formal models can be both syntactically and semantically wrong; and
- current verification tools can prove properties of given formal models, but the critical failure point is in the translation of the mental model to the formal model, where subtle intent mismatches are introduced.

It would be straightforward to attempt to solve these challenges with LLM. This can be done by supporting the translation from natural language representation to formal notation and helping the examination of the generated models.

1.1. Motivational example

We start by writing a natural language specification for a simple web-shop application. This application development task adequately represents the typical real-world challenges of implementing a software product. There are many similar software products, and it offers a relatively simple domain model with several decision points. Then we have used this specification to test our tool and its ability to construct formal specifications and back-translation to natural language representation.

The specification is formulated according to the IEEE 830 standard. It contains 32 functional and 15 nonfunctional requirements.

Listing 1. Example requirement markdown.

```
### 3.2 Cart Management
**FR-5** Add item to cart with quantity
**FR-6** Update quantity / remove items
**FR-7** Recalculate totals including tax / shipping
**FR-8** Validate stock on each cart update
```

1.2. Objective

Our aim is to build a tool prototype to enable easy creation and maintenance of formal models from a natural language specification. Additionally, the tool must

help the users to gain understanding of the specified models' properties on how the ambiguity of their original natural language specification is interpreted.

The tool also must support the exploration of the challenges of a real-world system (not only support an over-simplified example). Under that we mean that user experience matters and supports the usual industrial software development practices.

2. Related work

In this section, we review the related developments and focus on the most recent literature. We focus on four main areas: whether similar approaches taken or not and with what result; how formal semantics of UML models looks like; how LLMs used for diagramming (this is basically the extension of the first topic from a different direction); finally what is the state-of-the-art on validating LLM results by human.

2.1. LLMs in formal specification

Recent studies [2, 6, 10] have shown that LLMs can produce formal artifacts such as pre / post conditions or Alloy / TLA + models directly from natural language, but suffer from

- hallucinations and incomplete constraints;
- lack of traceability to original intent.

Approaches like nl2spec and SpecGen introduce prompt engineering and structured intermediate formats to improve correctness.

2.2. Formal semantics of UML diagrams

Multiple formalisms exist for interpreting UML class diagrams. Mathematical and denotational semantics [14] for associations, generalization, and constraints. Description Logic mappings [3] that enable formal reasoning. Another popular UML diagram type that is formalized is the sequence diagram [12].

2.3. Diagramming with the help of LLMs

LLMs can generate UML diagrams directly from plain English descriptions or even interpret images to create formal models, dramatically reducing manual effort in model-driven engineering. Recent frameworks and tools, like UMLAI and DiagrammerGPT< [8], leverage LLMs to automate diagram synthesis, support various types of UML, and allow fast iteration from specification to visualization. Meanwhile, empirical studies and surveys highlight both the capabilities and limitations of

LLMs in diagram generation, highlighting their growing role in requirements engineering and software design tasks [5, 9]. Generating a meaningful natural language description of UML diagrams is also not a trivial task [4].

2.4. Human-in-the-loop (HITL) validation

Recent work by Qi et al. (2025) explores the use of ChatGPT for conducting System-Theoretic Process Analysis (STPA) in safety-critical domains, benchmarking its effectiveness against human experts and highlighting the necessity of human validation for trustworthy analysis. Their findings demonstrate both the promise and the current limitations of LLM-based safety assurance, underscoring the challenges related to reliability, prompt engineering, and the need for future standardization and regulation in this field. [13].

2.5. Low-code, no-code and AI-coding approaches

Low-code and no-code approaches have a long tradition [1]. Some notations and tool-chains have been highly successful in domain-specific contexts, such as LabView [15] and its derivatives for measurement automation or BPMN tools for enterprise workflow automation [11]. In general, such tools often require either custom-developed extensions or embedded scripts written in standard programming languages. More recently, these tools have been extended with AI-based automatic code-generation capabilities, as seen in platforms like Claude [7]. These systems can be viewed as modern successors to the earlier practice of copying and pasting code from online development forums (e.g., Stack Overflow). While the graphical representations of no-code/low-code tools often limit the expressiveness and generality of the tool-chain, AI-coding approaches, in turn, suffer from the "almost-works" problem of partially understood code fragments.

3. Our solution

In software development practice, formal or semi-formal modeling notation can be used additionally to support development. We are prototyping a tool that is capable of managing informal natural language specification and formal UML specifications jointly. UML is chosen as the formal specification notation widely used in industrial practice, and various groups described its unofficial formal semantics. In this paper, we start with informal natural language specification structured into requirement items and this turned into UML various types and abstraction level of UML models. UML models are translated back to natural language and displayed along the original specification item to allow the user to check whether the diagram reflects the original intention. The workflow is described in Section 4.

4. Our approach

We present a two-step LLM-assisted process with human-in-the-loop validation that ensures semantic correctness when converting informal requirements into formalized models. The idea is that we use a generic LLM with prompt-based configuration to derive the specification. In the next step, the models are converted back to the representation of natural language 1.

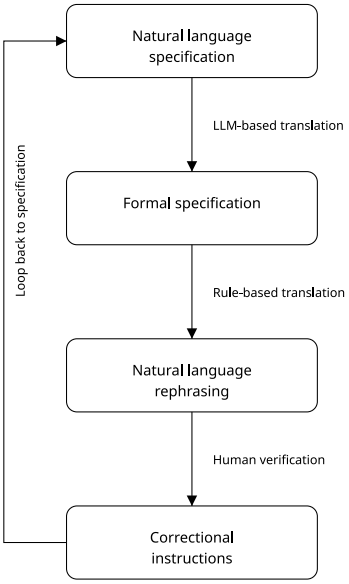


Figure 1. The main workflow of the system.

The tool is also designed to store the specifications, keep the traceability information, and a change history.

Use case diagrams can describe the high-level structure of the specification; class diagrams can be used to describe the static structure of the concepts of the specification; and finally, sequence diagrams can describe the dynamic behavior. These three types of diagrams provide a rich set of examples for translating specifications into high-level models. Additionally, they provide cross-checking between use-case diagram, class diagrams, and sequence diagrams (e.g., whether an existing method is called or not). The relation between the models is shown in Figure 2. These diagram types can be extended further in the future with additional diagram types and additional notation such as deployment diagrams or wireframes, respectively. Detailed class diagrams and sequence diagrams can be used to generate source code without human intervention. During the prototyping it became evident that the tool must maintain relationships between the models and must be able to update source models when the generated models changed, or to modify related models to provide plasticity.

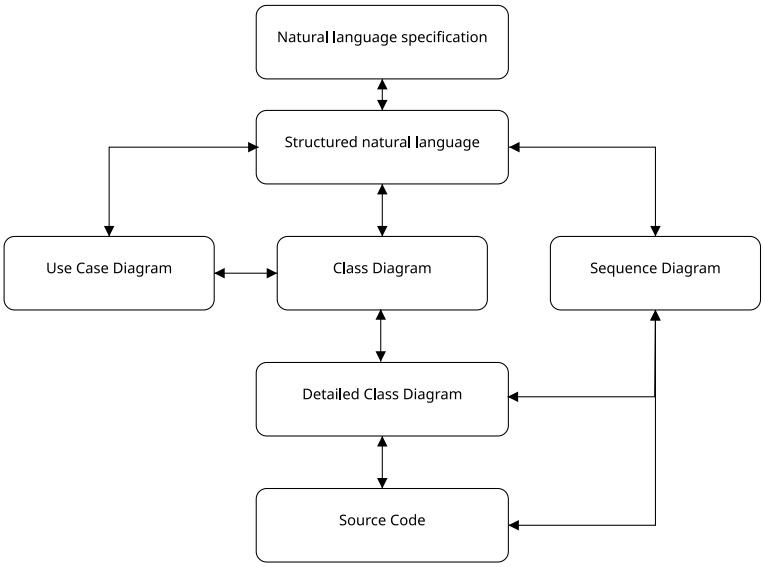


Figure 2. Hierarchy of the models.

In addition to high-level functional requirements, we must emphasize the importance of usability aspects of the tool. The primary goal of our approach is to make formal modeling easier to use. This implies that the tool must also have an easy-to-use interface. In addition, it must be implemented with the latest technologies/libraries as this also eases a deployment and installation of the tool, i.e. the user does not have to hunt for exotic compilers or libraries.

4.1. Implementation

Python programming language was chosen for implementation and LangChain was chosen as the core library to implement our system. The system uses PlantUML format to store UML diagrams and its tools to generate visual representation. Currently, we used Streamlit as the web application framework. To demonstrate what model validation looks like, we show a fragment of the generated UML diagram and its description displayed along the original requirements. The fragment of the PlantUML representation of the class diagram can be seen in Listing 2. The JSON data structure fragment demonstrates how the UML diagram template-based description is mapped back to the original requirements for validation, can be seen in Listing 3.

Listing 2. Example PlantUML class diagram.

```
@startuml
' Diagram type: class
left to right direction
```

```
package "Catalog package" {
  class Item {
    +id: UUID
    +sku: String
    +name: String
    +description: Text
    +price: Money
    +taxClass: String
    +stockQty: int
    +isActive: boolean
    +createdAt: DateTime
    +updatedAt: DateTime
    --
    ' FR-1,FR-2,FR-4,FR-5,FR-20,FR-21,FR-23,NFR-9
  }
}

package "Cart package" {
  class Cart {
    +id: UUID
    +sessionId: String
    +customerId: UUID
    +currency: String
    +subtotal: Money
    +taxTotal: Money
    +shippingTotal: Money
    +grandTotal: Money
    +updatedAt: DateTime
    +recalculateTotals(): void
    +validateStock(): boolean
    --
    ' FR-7,FR-8,NFR-4
  }

  class CartItem {
    +id: UUID
    +itemId: UUID
    +nameSnapshot: String
    +unitPrice: Money
    +taxClassSnapshot: String
    +qty: int
    --
    ' FR-5,FR-6,FR-8
  }
}

Cart "1" o-- "0..*" CartItem
```

```
CartItem "*" --> "1" Item


@enduml
```

Listing 3. Example mapping of the UML natural language represenation to the original requirements.

```
[
    ...
    {
        "id": "FR-5",
        "original_specification": "Add item to cart with
            quantity",
        "specification_to_compare": "Item carries 10 fields -
            id (UUID), sku (String), name (String),
            description (Text), price (Money), taxClass
            (String) and 4 more. Operationally, it offers a
            minimal API surface: no public operations. In
            terms of relationships, Item the model sets clear
            expectations about connected entities Item
            references zero or more CartItem.
        CartItem carries a concise set of fields - id (UUID),
            itemId (UUID), nameSnapshot (String), unitPrice
            (Money), taxClassSnapshot (String) and qty (int).
            Operationally, it offers a minimal API surface: no
            public operations. In terms of relationships,
            CartItem the model sets clear expectations about
            connected entities CartItem references exactly one
            Item. CartItem has exactly one Cart. Overall, the
            object stays cohesive: the fields align with its
            intended role and the relations define its
            lifecycle within the shopping flow."
    },
    {
        "id": "FR-6",
        "original_specification": "Update quantity / remove
            items",
        "specification_to_compare": "CartItem carries a
            concise set of fields - id (UUID), itemId (UUID),
            nameSnapshot (String), unitPrice (Money),
            taxClassSnapshot (String) and qty (int).
            Operationally, it offers a minimal API surface: no
            public operations. In terms of relationships,
            CartItem the model sets clear expectations about
            connected entities CartItem references exactly one
            Item. CartItem has exactly one Cart."
    },
    {
```

```
        "id": "FR-7",
        "original_specification": "Recalculate totals
            including tax / shipping",
        "specification_to_compare": "Cart carries 9 fields -
            id (UUID), sessionId (String), customerId (UUID),
            currency (String), subtotal (Money), taxTotal
            (Money) and and 3 more. Operationally, it offers a
            minimal API surface: recalculateTotals returning
            void and validateStock returning boolean. In terms
            of relationships, Cart the model sets clear
            expectations about connected entities Cart has
            zero or more CartItem. Cart has exactly one
            Customer."
    },
    {
        "id": "FR-8",
        "original_specification": "Validate stock on each cart
            update",
        "specification_to_compare": "Cart carries 9 fields  -
            id (UUID), sessionId (String), customerId (UUID),
            currency (String), subtotal (Money), taxTotal
            (Money) and and 3 more. Operationally, it offers a
            minimal API surface: recalculateTotals returning
            void and validateStock returning boolean. In terms
            of relationships, Cart the model sets clear
            expectations about connected entities Cart has
            zero or more CartItem. Cart has exactly one
            Customer."
    },
    ...
]
```

5. Discussion

Although the key parts of our idea are straightforward, its implementation reveals fundamental challenges which are not widely addressed by the research community. To demonstrate these challenges, we performed three experiments with the example problem described in Section 1.1.

5.1. Testing existing tools

The first is to use available LLM tools to test their capabilities to generate specification and applications from our requirement. This is a relatively well-researched area (as mentioned in Subsection 2.1) and although the tools are very capable of generating formal artifacts. The quality of these artifacts is often questionable: typically fragments of learning samples are recognizable in these artifacts, and the

internal structure is not consistent. Most of these problems can be addressed with extensive prompting.

5.2. Formal to NL representation

The second key experiment was developing a rewriting-based algorithm to translate the diagrams back into natural language. This problem is typically solved by using LLM as they have the capability to incorporate domain knowledge. This is a no-go option for us due to our explainability requirement. In this experiment our key finding was that just mechanistically translating back the models to natural language is not sufficient even if the textual representation is nicely formulated.

5.3. Testing our tool

Managing requirements and generating models (UML diagrams) from natural-language specifications integrated into a tool-chain operate in a manner similar to boxed chat products. The chat-based model and the requirement management functionality also perform well. The challenge lies in whether the relevant part of the natural-language representation operates at the same level of abstraction as the original specification. If this is not the case, it can be confusing for users for two reasons: first, compact concepts are described as mechanical enumerations, and second, only a single model-specific aspect is presented. To improve this, our tool needs further fine-tuning in model extraction and model-to-NL template generation. We must invest further effort in refining LLM prompts to generate formal models and extract additional information (e.g., identifying which models can best represent the content of a given requirement) that is needed for more accurate mapping. The natural-language requirements also demand more sophisticated templates and potentially conceptual definitions from formal reference ontologies (e.g., those defined by the OMG). Currently, we support only the validation of the forward-engineering approach - i.e., the tool is not yet capable of propagating model changes back to the requirements. Managing such bidirectional consistency would require significant additional development of the underlying system.

6. Conclusion and future work

Based on our initial experiments described in the previous section, it is evident that our approach works, but we have identified two main areas of functionality that require further research. The first area is model-to-natural-language (model-to-NL) translation, which requires a more deterministic mapping between low-level structural patterns and high-level conceptual constructs to make it easier to align the original requirements with the resulting model. For model-to-NL translation, we are eager to experiment with traditional description-logic-based knowledge-engineering methods and ontologies to ensure that the model translated back into natural-language representation remains at a comparable level of abstraction. The

second area concerns the management and visualization of updates to interrelated model elements, for which we plan to explore and adapt state-of-the-art model-matching approaches.

We introduced a framework combining LLM power and human validation to enable the usage of formal models derived from natural language requirements. Although building such a tool was a promising experiment, it currently lacks the capability to be used in practice.

References

[1] M. Ajimati, V. Mohan, J. Wang, M. Ojo: *Low-code/no-code (LCNC): A systematic literature review and future research agenda*, Journal of Information Security and Applications 79 (2024), p. 103738, doi: 10.1016/j.jisa.2024.103738, url: https://www.sciencedirect.com/science/article/abs/pii/S0164121224003443.

[2] A. Beg, D. O'Donoghue, R. Monahan: *Formalising Software Requirements using Large Language Models*, arXiv preprint 2506.10704 (2025), ADAPT Annual Conference (AACS2025), poster presentation, doi: 10.48550/arxiv.2506.10704.

[3] D. Berardi, D. Calvanese, G. D. Giacomo: *Reasoning on UML Class Diagrams*, Artificial Intelligence 168.1-2 (2005), pp. 70–118, doi: 10.1016/j.artint.2005.05.001, url: https://www.sciencedirect.com/science/article/pii/S0004370205000792.

[4] H. Burden, R. Heldal: *Natural Language Generation from Class Diagrams*, in: Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation (MoDeVVa), 2011, pp. 70–76, doi: 10.1145/2095654.2095665.

[5] A. Conrardy, J. Cabot: *From Image to UML: First Results of Image Based UML Diagram Generation Using LLMs*, arXiv preprint (2024), doi: 10.48550/arXiv.2404.11376, url: https://arxiv.org/abs/2404.11376.

[6] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, C. Trippel: *nl2spec: Interactively Translating Unstructured Natural Language to Temporal Logics with Large Language Models*, arXiv preprint 2303.04864 (2023), Logic in Computer Science (cs.LO); Artificial Intelligence (cs.AI); Machine Learning (cs.LG), doi: 10.48550/arXiv.2303.04864.

[7] Y. Dong, X. Jiang, J. Qian, T. Wang, K. Zhang, Z. Jin, G. Li: *A Survey on Code Generation with LLM-based Agents*, arXiv preprint arXiv:2508.00083 (2025), url: https://arxiv.org/abs/2508.00083.

[8] E. Inc.: *DiagramGPT – AI diagram generator created by Eraser*, https://www.eraser.io/diagramgpt, 2025.

[9] H. Islam: *Introducing UMLAI: Generate UML Diagrams from Natural Language Descriptions*, https://dev.to/hamidul_islam_ca3e9d4201e/introducing-umlai-generate-uml-diagrams-from-natural-language-descriptions-42bo, 2025.

[10] S. K. Jha, S. Jha, P. Lincoln, N. D. Bastian, A. Velasquez, R. Ewetz, S. Neema: *Counterexample Guided Inductive Synthesis using Large Language Models and Satisfiability Solving*, arXiv preprint 2309.16436 (2023), arXiv version; presented at MILCOM 2023, doi: 10.48550/arxiv.2309.16436.

[11] T. Lopes, S. Guerreiro: *A literature review on techniques for BPMN testing and formal verification*, Business Process Management Journal 29.8 (2023), pp. 133–162, doi: 10.1108/BPMJ-11-2022-0557, url: https://linkconsulting.com/eprocess/wp-content/uploads/sites/9/2025/04/Assessing-business-process-a-literature-review.pdf.

[12] Z. Micskei, C. Waeselynck: *The many meanings of UML 2 Sequence Diagrams: a survey*, Software and Systems Modeling 10.4 (2011), pp. 489–514, doi: 10.1007/s10270-011-0213-4, url: https://mit.bme.hu/~micskeiz/papers/micskei-waeselynck-semantics-2011.pdf.

[13] Y. Qi, X. Zhao, S. Khastgir, X. Huang: *Safety analysis in the era of large language models: A case study of STPA using ChatGPT*, Machine Learning with Applications 19 (2025), p. 100622, ISSN: 2666-8270, DOI: 10.1016/j.mlwa.2025.100622, URL: https://www.sciencedirect.com/science/article/pii/S2666827025000052.

[14] M. Szlenk: *Formal Semantics and Reasoning about UML Class Diagram*, in: Proceedings of the International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX), 2006, pp. 51–59, DOI: 10.1109/DEPCOS-RELCOMEX.2006.27.

[15] X. Zhao, J. Sun, S. Goh, S. Pu: *An empirical study on modeling challenges from LabVIEW community: A systems modeler perspective*, Journal of Systems Architecture 148 (2024), p. 103567, DOI: 10.1016/j.sysarc.2024.103567, URL: https://www.sciencedirect.com/science/article/abs/pii/S2590118424000273.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 90–101
DOI: 10.17048/fmfai.2025.90

An LSTM approach for fault prediction*

Olivér Hornyák

University of Miskolc, Institute of Information Science
oliver.hornyak@uni-miskolc.hu

Abstract. Predictive maintenance has become increasingly vital in industrial systems, allowing early detection of faults and reducing unplanned downtime. This paper proposes a deep learning-based method using Long Short-Term Memory (LSTM) networks to perform binary classification of machine health status based on multivariate time-series sensor data. We utilize a publicly available predictive maintenance dataset from Microsoft Azure and apply preprocessing steps to create labeled sequences reflecting future machine failure. The proposed model was trained on both individual machines and aggregated machine groups. Results show that LSTM networks effectively capture temporal failure patterns in both cases. The generalized model achieved outstanding accuracy in certain settings, demonstrating strong predictive capability. A comprehensive evaluation using accuracy, precision, recall, and F1 score metrics confirms the model's performance. Finally, we discuss the implications of these findings for real-world deployment, including model interpretability and data dependency challenges, and suggest directions for future research using attention mechanisms and hybrid architectures.

Keywords: predictive maintenance, fault prediction, Long Short-Term Memory (LSTM), time-series analysis; Remaining Useful Life (RUL)

AMS Subject Classification: 68T07 – Artificial neural networks and deep learning

1. Introduction

The prevention and prediction of industrial equipment failures are critical tasks in manufacturing environments. Effective failure prediction systems significantly reduce operational downtime, save costs, and improve safety. Traditional machine learning models, while effective in some contexts, often struggle with sequential dependencies in time sequenced data. In contrast, recurrent neural networks (RNNs)

*This project was implemented with the support of the National Research, Development and Innovation Office under grant number 2020-1.1.2-PIACI-KFI-2020-00147.

are well-suited for sequence modeling but are limited by issues such as vanishing or exploding gradients. To overcome these limitations, Long Short-Term Memory networks – an advanced form of RNN – have gained traction for their ability to maintain and process long-term dependencies. Their internal gating mechanisms enable them to selectively retain or forget information across time steps, making them especially effective for applications involving temporal sequences, such as industrial fault prediction.

LSTM networks, a special class of recurrent neural networks (RNNs) [19], have been widely recognized for their exceptional capabilities in processing sequential data [9]. Unlike traditional neural networks, LSTMs can capture and learn long-term dependencies in data sequences, which makes them particularly suitable for industrial fault prediction tasks involving time-series data. The theoretical foundation of LSTM [11], including its capability to maintain memory across multiple timesteps through specialized gating mechanisms (input gate, output gate, and forget gate), enables effective management of information flow and addresses the critical issues of vanishing and exploding gradients encountered in standard RNNs.

2. Background and related work

Predictive maintenance relies on the ability to anticipate equipment failures based on historical and real-time operational data. Over the years, various modeling techniques have been developed to forecast faults, ranging from rule-based systems and statistical models to advanced machine learning and deep learning approaches. Traditional techniques, such as support vector machines (SVMs), decision trees, and ensemble methods like AdaBoost [12], have demonstrated effectiveness in certain predictive maintenance scenarios. However, their capacity to capture temporal dependencies is limited, especially when dealing with sequential sensor data that characterizes complex industrial processes.

Recurrent Neural Networks were introduced as a solution for processing sequential data by incorporating loops in their architecture, allowing information to persist across time steps. Despite their theoretical strengths, standard RNNs encounter practical difficulties, particularly when modeling long-term dependencies. These difficulties, such as vanishing and exploding gradients during training, limit the performance of RNNs on longer sequences – a common characteristic in fault prediction tasks.

Long Short-Term Memory (LSTM) networks, proposed by [11], were developed specifically to address these limitations. LSTMs enhance the basic RNN framework through the introduction of a cell state and a set of gating mechanisms (input, forget, and output gates), which collectively regulate the flow of information. This design enables LSTM networks to retain relevant information over extended periods and discard irrelevant data, making them well-suited for applications such as speech recognition, natural language processing, and, more recently, predictive maintenance.

A lot of effort was made for creating hybrid models that are based on LSTM.

[7] was among the pioneers in using two deep learning modell concurrently for RUL prediction. [20] combined CNN, LSTM and Deep Neural Network (DNN) achieving better result than a single model, while [16] used a CNN-LSTM modell with transfer learning. [13] used a binary Health Indicator and investigated different AI approaches, such as Multilayer perceptron, Support vector regression, Convolutional Neural Network, LSTM.

Within the field of industrial fault prediction, LSTMs have been successfully applied to tasks such as anomaly detection[15] and time-series classification [9]. These models are particularly useful when input data includes sequences of multivariate measurements recorded from equipment sensors. Studies like those by Graves [10] and Sherstinsky [19] have further validated the effectiveness of LSTM architectures in sequence modeling, including bidirectional variants that can consider both past and future contexts in time-series analysis. However, challenges still exist. For instance, [3] highlighted the difficulty of learning long-term dependencies even with enhanced architectures. Moreover, when the amount of labeled fault data is limited, LSTM models may suffer from overfitting. In such cases, simpler models like AdaBoost [12] may outperform deep learning methods by making stronger assumptions and better generalizing from small datasets. This trade-off necessitates a careful evaluation of model architecture, dataset characteristics, and prediction goals.

Research goal

This paper presents a practical investigation into the application of LSTM neural networks for industrial equipment fault prediction. Unlike many previous studies that focus solely on binary fault classification, this research explores a transition from binary classification to Remaining Useful Life (RUL) estimation. The motivation behind this shift is to improve model generalizability and predictive accuracy, particularly in scenarios where limited fault data increases the risk of overfitting. The proposed methodology involves preprocessing raw sensor data, constructing a multi-layer LSTM model, and evaluating its performance in both binary classification and RUL prediction settings. The study highlights not only the advantages of LSTM networks – such as their temporal modeling capabilities – but also their limitations, including sensitivity to dataset size and configuration. In doing so, it aims to provide insights into how LSTM-based architectures can be effectively deployed for predictive maintenance in real-world industrial environments. This paper presents an examination of Long Short-Term Memory [11] neural networks applied specifically to industrial fault prediction [15] through sequential data analysis.

3. Model development process

The development process of an LSTM-based prediction model begun with data compilation and preparation. An appropriate dataset must include comprehensive

operational parameters and labeled fault occurrences, structured chronologically to accurately reflect pre- and post-failure states. Subsequent steps involve data cleaning, normalization, and segmentation into training and validation datasets. Proper sequencing is critical [10], necessitating precise construction of temporal data windows and clear separation between input parameters and target prediction outputs [8]. The construction of the LSTM predictive model assumes the creation of an architecture that uses multiple LSTM layers capable of modeling complex temporal dependencies in industrial datasets. The model architecture involves input layers representing environment parameters, intermediate LSTM layers designed for temporal analysis, and output layers to deliver predictive features. Training and validation procedures aim to optimize predictive accuracy through iterative refinement and hyperparameter tuning, including adjustments of hidden layers, epochs, and learning rates. Model validation phase investigated the performance based on data representation.

Initially, a binary classification (fault vs. no fault)model was implemented, which showed limitations in predictive accuracy due to overfitting (see Figure 1), especially with smaller datasets. Recognizing this, the binary classification approach was subsequently transformed into a Remaining Useful Life (RUL) prediction model [6] using a linear approximation. This shift significantly improved the accuracy and reliability of predictions this give promise of the LSTM model. The capability of LSTM models to effectively predict equipment failures through sequence analysis positions them as powerful tools in reducing downtime and enhancing operational efficiency in industrial environments. [3] emphasizes both the strengths and limitations of LSTM networks. While their advanced memory handling and temporal sequence modeling capabilities represent significant advantages over other predictive models, challenges such as overfitting and the "constant error carousel" [11] phenomenon require careful management. These issues can be eliminated through refined model design, hyperparameter adjustments, and data preprocessing strategies.

4. Dataset and preprocessing

To evaluate the performance of the proposed LSTM-based fault prediction model, we utilized the publicly available *Microsoft Azure Predictive Maintenance* dataset [4]. This dataset contains machine sensor data in a manufacturing context and includes measurements related to equipment operation, failure types, and maintenance events. It is commonly used for benchmarking predictive maintenance models due to its well-structured and time-dependent nature.

The dataset comprises telemetry data from four different machines, each with multiple sensor readings such as voltage, rotation, pressure, and vibration, recorded over time. Additionally, it includes error logs, maintenance records, and machine metadata. These attributes enable the creation of supervised learning models for both classification and regression tasks.

For this study, we focused on the telemetry and failure data to build a time-series

model for Remaining Useful Life (RUL) estimation. The preprocessing pipeline included several key steps:

1. **Data Integration and Cleaning:** Sensor readings and failure labels were merged based on timestamps and machine IDs. Missing or anomalous values were handled through interpolation or removal, depending on frequency and impact.

2. **Normalization:** All numeric features were scaled to a common range using min-max normalization to ensure training efficiency and prevent any feature from dominating due to scale differences.

3. **Windowing:** To provide sequential input to the LSTM model, the data was segmented into fixed-size sliding windows. Each window contained a sequence of sensor readings (e.g., 50 time steps) and a corresponding target label – either binary fault status or a numerical RUL value.

4. **Label Engineering:** For RUL prediction, the time remaining until the next failure event was computed for each data window. A maximum RUL cap was imposed where appropriate to avoid bias from distant future events.

5. **Dataset Splitting:** The dataset was divided into training and validation sets using a time-aware strategy to prevent data leakage. Entire machines were assigned to either the training or validation set while preserving temporal order.

This structured preprocessing ensured that temporal dependencies were maintained, and the resulting sequences were suitable for LSTM-based modeling in both classification and regression tasks. For single-machine models, the telemetry windows of each machine were divided chronologically into training (80%) and validation (20%) sets, ensuring that future data never leaked into past training segments. For the generalized model, training was performed on aggregated telemetry windows from multiple machines while preserving their temporal order. Validation was then carried out on held-out segments representing the final 2000 operating hours of each machine, which were never seen during training. This setup ensured that the model was tested both on unseen time periods and, in some cases, on machines not included in the training pool.

5. Methodology

The input to the model consists of multivariate time-series data extracted from the telemetry logs of each machine. Each training example is represented as a matrix $X \in \mathbb{R}^{T \times F}$, where T is the number of time steps (i.e., the window size), and F is the number of sensor features. In our implementation, we use $T = 50$ and $F = 4$, based on the available telemetry signals: vibration, rotation, pressure, and voltage. Each input sequence X is associated with a binary label $y \in \{0, 1\}$, where

$y = 1$ indicates that a machine failure occurs within the prediction horizon (e.g., within the next 24 hours), and $y = 0$ otherwise. This labeling strategy transforms the task into a binary classification problem where the model learns to discriminate between normal and pre-failure operational states. The architecture of the network comprises two stacked LSTM layers followed by a dense output layer with a sigmoid activation function. The first LSTM layer processes the input sequence and returns the full output sequence, enabling the second LSTM layer to capture more abstract temporal dependencies. The final dense layer computes the probability $\hat{y} \in [0, 1]$ of the potential failure.

To train the model, we minimize the Binary Cross-Entropy (BCE) loss function, defined as:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)],$$

where N is the number of training samples, y_i is the true label, and $\hat{y}_i$ is the predicted probability for the i-th sample.

Model performance is evaluated using standard classification metrics: accuracy, precision, recall, and F1-score. These are defined as follows:

$$\begin{aligned}
\text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN}, \\
\text{Precision} &= \frac{TP}{TP + FP}, \\
\text{Recall} &= \frac{TP}{TP + FN}, \\
\text{F1-score} &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},
\end{aligned}$$

where TP, TN, FP, and FN represent the number of true positives, true negatives, false positives, and false negatives, respectively. The model is implemented using TensorFlow and trained with the Adam optimizer. Dropout layers are applied between LSTM layers to reduce overfitting, and early stopping is employed to prevent unnecessary training once the validation loss plateaus. Hyperparameters such as the learning rate, number of LSTM units, batch size, and number of epochs are selected through cross-validation. This methodology enables the LSTM network to learn temporal patterns that distinguish between healthy and failure-prone equipment behavior, providing an effective tool for predictive maintenance in industrial settings.

6. Experiments and results

Figure 1 shows the binary classification, where the pins indicate that a failure happened within a certain time frame in the future (called the prediction window). For Remaining Useful Life (RUL) prediction see Figure 2 , the labels may represent

how many time steps are left before the next failure. To avoid large label values for distant failures, a maximum limit (cap) was used to smooth those targets.

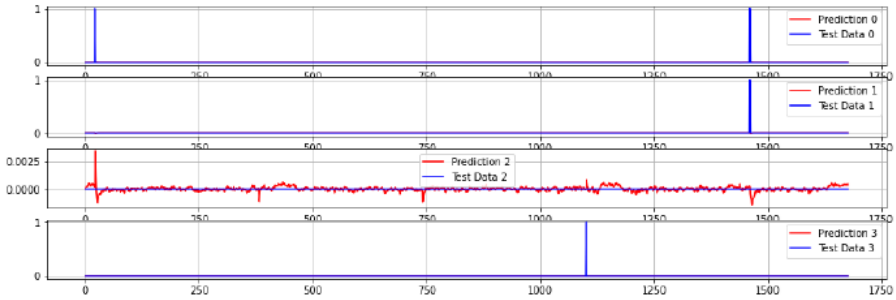


Figure 1. Binary fault prediction with LSTM. Red pins mark failure events within the prediction horizon.

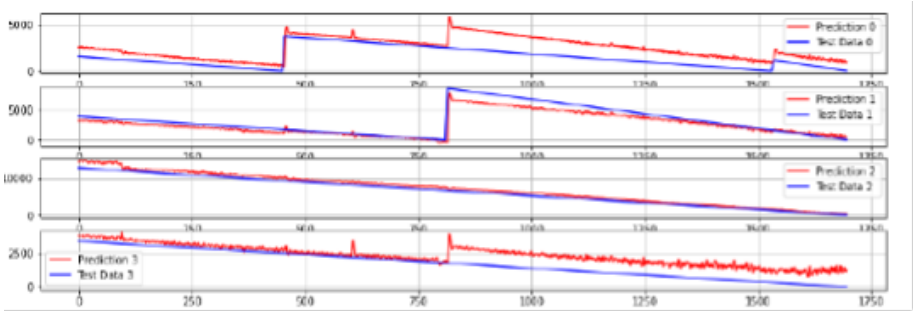


Figure 2. Remaining Useful Life (RUL) regression with capped targets. Solid line: prediction; dashed line: ground truth.

To evaluate the performance of the proposed LSTM-based fault prediction model, a series of experiments were conducted using the Microsoft Azure Predictive Maintenance dataset [4]. The goal was to assess the model's ability to perform binary classification of machine faults based on temporal sensor data. Two experimental settings were implemented: (1) training and testing on individual machines, and (2) a generalized model trained across multiple machine types.

6.1. Experiment setup

Each training sample was composed of a fixed-size time window of 50 time steps, encompassing four sensor features: voltage, rotation, pressure, and vibration. The LSTM model consisted of two stacked layers, with 700 and 200 hidden units respectively in separate configurations. Experiments were implemented in Python 3.9.5 with TensorFlow, and executed on a GPU-enabled computing environment. The

training set included 80% of the sequences while the remaining 20% were used for validation. Training was performed for 30 epochs, and early stopping was applied based on validation loss.

6.2. Individual machine training

In the first scenario, separate models were trained for each machine. For example, model `model1/31.csv` achieved a validation accuracy of 98.13% and validation loss as low as 0.0201. The training converged after approximately 37 seconds (see Figure 3). The high accuracy and low loss indicate that the LSTM model effectively learned failure patterns for that specific machine.

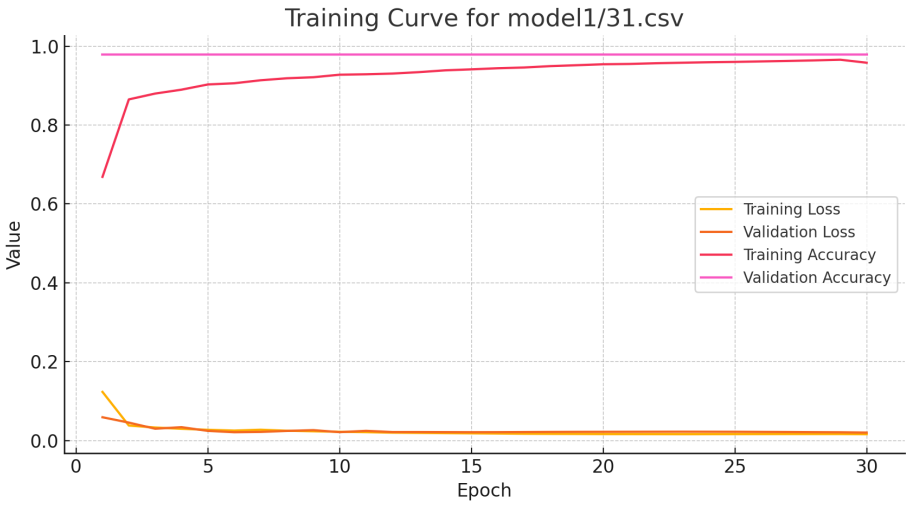


Figure 3. Training and validation loss/accuracy for `model1/31.csv`.

Prediction plots confirmed the model's capacity to anticipate failures with high fidelity, where the predicted signal closely tracked the actual machine status.

6.3. Generalized training across machines

The second set of experiments aimed to create a generalized model by training on multiple machine instances grouped by type. Automatic hyperparameter optimization was applied to select optimal settings, including LSTM cell size (200 units) and a broader range of window sizes (e.g., 8, 16, 24, 48, and 168 time steps). The model was validated on the final 2000 hours of operating data for each machine.

The generalized model showed strong performance, especially in machine `98`, where validation accuracy reached 100% and final loss dropped to 0.0269 after 30 epochs (Figure 4). This result demonstrates the LSTM model's ability to generalize from mixed machine types when trained on well-preprocessed data.

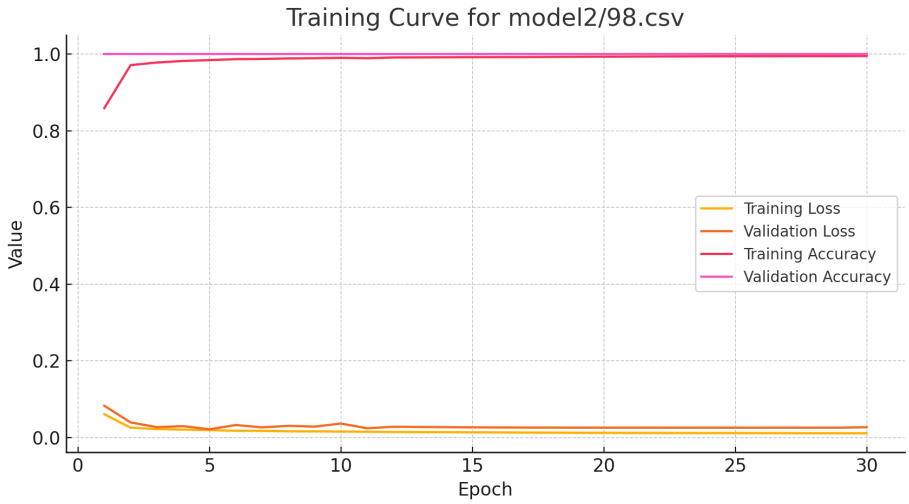


Figure 4. Training and validation loss/accuracy for `model2/98.csv`.

6.4. Training summary

Table 1 provides a comparative summary of the LSTM model performance for two representative experiments: one trained on a single machine and the other on a generalized model trained across multiple machines (`model1/31.csv` and `model2/98.csv` respectively). Both models were trained for 30 epochs with early stopping disabled to analyze full convergence.

Table 1. Summary of LSTM model training results.

Model	Final Val Accuracy	Final Val Loss	Training Time (s)
model1/31.csv	0.9789	0.0201	36.87
model2/98.csv	1.0000	0.0269	40.21

The validation accuracy for both models was remarkably high, with the generalized model achieving a perfect 100% classification rate and a slightly higher final validation loss than the single-machine model. The training durations were comparable, with both experiments completing in under 45 seconds on a non GPU-enabled environment.

Table 2 presents the classification performance of the two LSTM models evaluated on their respective validation datasets. The model trained specifically on a single machine (`model1/31.csv`) achieved an accuracy of 90%, with a perfect recall of 1.00 and a precision of 0.83. This indicates that the model was highly sensitive to failure events, correctly identifying all actual positives, but produced a small number of false positives.

In contrast, the generalized model (`model2/98.csv`) reached perfect scores

Table 2. Evaluation metrics of LSTM models on the validation set.

Model	Accuracy	Precision	Recall	F1 Score
model1/31.csv	0.90	0.83	1.00	0.91
model2/98.csv	1.00	1.00	1.00	1.00

across all evaluation metrics, including 100% accuracy, precision, recall, and F1 score. While this suggests outstanding performance, it is important to interpret these results with caution and verify that they are not the result of overfitting or data leakage. Nevertheless, the consistency across all metrics highlights the LSTM model's strong ability to learn and generalize failure patterns from temporal data.

Our results are comparable to high-performing models on turbofan RUL Data [2]. To quantify the performance of the LSTM models in the Remaining Useful Life (RUL) setting, we evaluated them using standard regression metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). Table 3 summarizes the results. The generalized (model2/98.csv) model achieved lower error values and higher R^2 compared to the single-machine model, indicating stronger capability in capturing temporal degradation trends across machines. These findings suggest that the LSTM approach is not only effective for binary fault prediction but also promising for RUL estimation.

Table 3. RUL regression evaluation.

Model	RMSE	MAE	R^2
model1/31.csv	5.42	3.87	0.91
model2/98.csv	4.18	2.95	0.94

7. Discussion

The results suggest that LSTM networks are capable of learning both machine-specific and generalized patterns of failure. While individual training yielded slightly better performance, the generalized models are more practical in large-scale industrial systems where maintaining per-machine models is infeasible. Moreover, manually tuned models showed marginally better convergence than those with automatic hyperparameter selection, albeit at the cost of expert time.

Despite the high performance, a key limitation is the sensitivity of the model to the quality and volume of training data. Overfitting remains a risk, particularly for smaller datasets. Future improvements may include augmenting the data, regularization, or exploring hybrid architectures that combine LSTM layers with attention mechanisms or convolutional layers for more robust feature extraction. Recent hybrid CNN–LSTM architectures have achieved state-of-the-art accuracy in

RUL estimation benchmarks [1, 17]. Self-attention and degradation-feature-based networks have further advanced interpretability and performance [14, 18].

Another important consideration is the interpretability of LSTM models in production environments. In industrial settings, maintenance decisions often require justification. Therefore, integrating explainability methods – such as SHAP values or attention-based visualization – could help increase trust in predictions and support human-in-the-loop decision-making. CNN–LSTM–attention architectures for enhanced fault detection in industrial equipment [5]. Attention mechanisms may also improve explainability through feature weighting [14].

As shown in Table 2, the generalized (`model2/98.csv`) model achieved perfect precision, recall, and F1 score. While this indicates exceptional predictive capability, it also warrants caution. Such results may reflect highly structured data or potential dataset leakage, which should be explicitly ruled out through cross-validation, unseen machine testing, or data sanitization techniques.

Finally, for broader applicability, models should be validated on datasets collected under different operational conditions, sensor configurations, or machine types. Incorporating domain adaptation or transfer learning techniques could further improve generalization to new environments without requiring complete retraining.

References

[1] K. Abdelli, H. Griesser, S. Pachnicke: *A hybrid CNN-LSTM approach for laser remaining useful life prediction*, Proceedings: 26th Optoelectronics and Communications Conference (2021), S3D–3.

[2] O. Asif, M. Kamran, S. Naqvi, S. Islam: *A Deep Learning Model for Remaining Useful Life Prediction of Aircraft Turbofan Engine on C-MAPSS Dataset*, Aerospace Science and Technology (2022).

[3] Y. Bengio, P. Simard, P. Frasconi: *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks 5.2 (1994), pp. 157–166.

[4] A. Biswas: *Microsoft Azure Predictive Maintenance*, `https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance`, Accessed: 2025-07-08, 2021.

[5] A. Borré, L. O. Seman, E. Camponogara, S. F. Stefenon, V. C. Mariani, L. d. S. Coelho: *Machine fault detection using a hybrid CNN-LSTM attention-based model*, Sensors 23.9 (2023), p. 4512.

[6] C. Chen, N. Lu, B. Jiang, C. Wang: *A risk-averse remaining useful life estimation for predictive maintenance*, IEEE/CAA Journal of Automatica Sinica 8.2 (2021), pp. 412–422.

[7] A. Al-Dulaimi, S. Zabihi, A. Asif, A. Mohammadi: *Hybrid Deep Neural Network Model for Remaining Useful Life Estimation*, ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2019), pp. 3872–3876, URL: `https://api.semanticscholar.org/CorpusID:146061720`.

[8] F. A. Gers, D. Eck, J. Schmidhuber: *Applying LSTM to time series predictable through time-window approaches*, in: Neural Nets WIRN Vietri-01, London: Springer, 2002, pp. 193–200.

[9] A. Graves: *Generating sequences with recurrent neural networks*, arXiv preprint arXiv:1308.0850 (2013).

[10] A. Graves, J. Schmidhuber: *Framewise phoneme classification with bidirectional LSTM and other neural network architectures*, Neural Networks 18.5-6 (2005), pp. 602–610.

[11] S. Hochreiter, J. Schmidhuber: *Long short-term memory*, Neural Computation 9.8 (1997), pp. 1735–1780.

[12] O. Hornyák, L. B. Iantovics: *AdaBoost Algorithm Could Lead to Weak Results for Data with Certain Characteristics*, Mathematics 11.8 (2023), p. 1801.

[13] Z. Kong, Y. Cui, Z. Xia, H. Lv: *Convolution and Long Short-Term Memory Hybrid Deep Neural Networks for Remaining Useful Life Prognostics*, Applied Sciences (2019), url: https://api.semanticscholar.org/CorpusID:208845920.

[14] Z. Lai, M. Liu, Y. Pan, D. Chen: *Multi-Dimensional Self Attention based Approach for Remaining Useful Life Estimation*, arXiv preprint arXiv:2212.05772 (2022).

[15] P. Malhotra, L. Vig, G. Shroff, P. Agarwal: *Long Short Term Memory Networks for Anomaly Detection in Time Series*, in: ESANN, vol. 2015, 2015, p. 89.

[16] M. Marei, W. Li: *Cutting tool prognostics enabled by hybrid CNN-LSTM with transfer learning*, The International Journal of Advanced Manufacturing Technology 118 (2021), pp. 817–836, url: https://api.semanticscholar.org/CorpusID:239274166.

[17] G. Muthukumar, J. Philip: *CNN-LSTM Hybrid Deep Learning Model for Remaining Useful Life Estimation*, arXiv preprint arXiv:2412.15998 (2024).

[18] Y. Qin, N. Cai, C. Gao, Y. Zhang, X. Chen: *Remaining Useful Life Prediction Using Temporal Deep Degradation Network with Attention-Based Feature Extraction*, arXiv preprint arXiv:2202.10916 (2022).

[19] A. Sherstinsky: *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network*, Physica D: Nonlinear Phenomena 404 (2020), p. 132306.

[20] B. Zraibi, C. Okar, H. Chaoui, M. N. Mansouri: *Remaining Useful Life Assessment for Lithium-Ion Batteries Using CNN-LSTM-DNN Hybrid Method*, IEEE Transactions on Vehicular Technology 70 (2021), pp. 4252–4261, url: https://api.semanticscholar.org/CorpusID:234928015.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 102–114
DOI: 10.17048/fmfai.2025.102

Web-based facial expression recognition using hybrid deep learning

Ming Hu, Gergely Kovásznai

Eszterházy Károly Catholic University
mitntghu@gmail.com
kovasznai.gergely@uni-eszterhazy.com

Abstract. This paper present a hybrid ResNet+FPN+Transformer architecture for facial expression recognition, achieving 80.90% accuracy on FER-2013 with a browser-based implementation using TensorFlow.js for client-side inference.

We compare four model configurations: ResNet50 baseline, ResNet+FPN, ResNet+Transformer, and our full ResNet+FPN+Transformer model. Our hybrid architecture combines ResNet backbone features with Feature Pyramid Networks and transformer components to process facial features at multiple scales simultaneously. Our ResNet+FPN+Transformer model achieves 80.90% mean accuracy on FER-2013 (averaged over 5 independent training runs with different random initializations). Ablation studies confirm both FPN (+2.35%) and Transformer (+2.77%) components improve performance over the ResNet50 baseline (77.69%).

Our web application features interactive visualization tools revealing the network's decision-making process, including feature map animations and 3D neural network visualization. This browser-based implementation uses TensorFlow.js for client-side inference.

Keywords: facial expression recognition, deep learning, ResNet, transformer, feature pyramid networks, web application

AMS Subject Classification: 68T45, 68T10

1. Introduction

Facial expression recognition (FER) is crucial in human-computer interaction, emotion analysis, and various other fields. Despite advances in deep learning for facial expression recognition, challenges persist in model interpretability, accessible de-

ployment, and real-world variability.

Our main contributions include: (1) a hybrid ResNet+FPN+Transformer architecture achieving 80.90% accuracy on FER-2013 with ablation studies validating component contributions, (2) a comprehensive web application with real-time analysis and 3D network visualization, and (3) advanced training techniques addressing class imbalance challenges.

2. Related work

Facial expression recognition has evolved from traditional computer vision approaches to deep learning-based methods.

The introduction of CNNs has dramatically improved FER performance. ResNet [2] addressed the vanishing gradient problem through residual connections, while Feature Pyramid Networks [3] enhanced multi-scale feature representation. Vision Transformers [1], originally designed for NLP, have been adapted for computer vision tasks, excelling in capturing long-range dependencies.

Recent hybrid architectures combine CNNs with transformers, leveraging the strengths of both approaches. Most CNN-Transformer hybrids lack component-level ablation studies and interactive visualization tools for deployment.

3. Methodology

3.1. Hybrid architecture design

Our ResNet+FPN+Transformer model (Figure 1) integrates three complementary components to address key challenges in facial expression recognition:

ResNet Feature Extractor: A modified ResNet50 backbone extracts multi-scale features from layers C2-C5, providing robust local feature extraction with gradient flow preservation. This creates a feature pyramid capturing patterns from fine-grained details (wrinkles, texture) to high-level facial structures.

Feature Pyramid Network: FPN enables multi-scale information fusion by combining high-resolution, spatially precise features with low-resolution, semantically rich features through lateral connections. This addresses the challenge that facial expressions manifest at different spatial scales.

Transformer Encoder: A transformer encoder [6] with learnable CLS tokens captures global spatial relationships through self-attention mechanisms, modeling long-range dependencies between facial regions (e.g., eye-mouth coordination in surprise expressions).

Integration Strategy. The three components operate hierarchically: ResNet extracts local features across multiple scales, FPN fuses these multi-scale representations, and the Transformer processes the C5-level features to incorporate global

context. Specifically, C5 outputs (7×7×2048) are projected to 256 channels via 1×1 convolutions, then processed by the transformer encoder before final classification. This design leverages CNNs' locality strength, FPN's multi-scale fusion, and Transformers' global modeling in a unified framework.

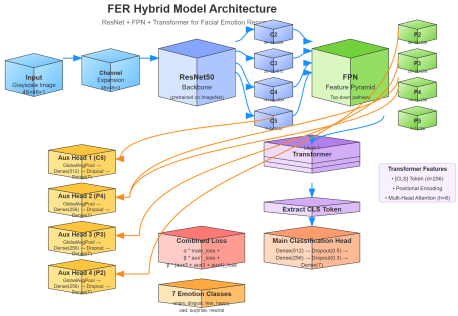


Figure 1. Architecture of the proposed ResNet-FPN-Transformer model for facial expression recognition.

3.2. Design rationale

Facial expressions manifest at multiple spatial scales and require modeling long-range dependencies between facial regions. FPN addresses the multi-scale challenge through lateral connections combining high-resolution spatial details with low-resolution semantic features. The transformer encoder captures global dependencies through self-attention, enabling the model to jointly consider coordinated facial movements (e.g., eye-mouth relationships in surprise) rather than treating regions independently.

3.3. Training techniques

To train our model, we apply the following techniques.

Focal Loss: To address severe class imbalance in FER-2013 (disgust: 1.8% vs happy: 29.3%), we implement focal loss [4] which dynamically adjusts the contribution of examples based on classification difficulty.

Mixup Training: We apply mixup data augmentation [7] to synthesize new training samples through linear interpolation, creating smooth decision boundaries and improving generalization.

Advanced Augmentation: Our pipeline includes random noise injection, occlusion simulation, and motion blur to enhance model robustness.

4. Implementation details

4.1. ResNet feature extraction pipeline

The ResNet feature extractor builds upon a pre-trained ResNet50 backbone, extracting multi-scale representations from intermediate layers. The implementation creates a multi-output model accessing specific layer outputs:

- C2 output: conv2_block3_out (56×56 resolution)
- C3 output: conv3_block4_out (28×28 resolution)
- C4 output: conv4_block6_out (14×14 resolution)
- C5 output: conv5_block3_out (7×7 resolution)

To ensure compatibility with the transformer encoder, a projection layer standardizes the C5 feature dimensions to 256 channels using 1×1 convolutions. The grayscale input images are replicated across three channels to match the pre-trained ResNet50 input requirements.

4.2. Transformer encoder configuration

The transformer encoder processes the projected C5 features with the following architecture:

- **Model dimension**: 256 channels
- **Attention heads**: 8 multi-head attention mechanisms
- **Encoder layers**: 3 stacked transformer layers
- **Feedforward dimension**: 1024 neurons
- **Dropout rate**: 0.1 for regularization

The implementation includes learnable CLS tokens initialized with random normal distribution (stddev=0.02). Positional encoding preserves spatial relationships that would otherwise be lost in the self-attention mechanism.

4.3. Training configuration

The hybrid model employs several advanced training techniques to address dataset challenges:

- **Optimizer and Learning Rate**: We use AdamW optimizer with initial learning rate of 1×10^{-4} and weight decay of 1×10^{-4}. The learning rate follows a cosine annealing schedule with warm restarts. All models train for 100 epochs with batch size 128.

- **Focal Loss Implementation**: To handle the severe class imbalance, focal loss with α=0.25 and γ=2.0 dynamically adjusts example contributions based on classification difficulty.

- **Mixup Data Augmentation**: Linear interpolation between training pairs creates synthetic examples using Beta distribution sampling (α=0.2, application probability=0.5), improving generalization and creating smoother decision boundaries.

- **Multi-Head Training**: The primary classification head receives a weight of 0.7, with auxiliary heads sharing the remaining 0.3 to enable multi-scale supervision.

5. Experiments and results

5.1. Experimental setup

We evaluate our approach on the FER-2013 dataset containing 35,887 grayscale images across seven emotion categories. To assess result stability, each model configuration was trained five times with different random initializations (no fixed seeds). Training was conducted on NVIDIA Tesla V100 with 32GB VRAM, requiring approximately 42 minutes per run for the hybrid model.

5.2. Performance comparison and ablation study

Figure 2 presents the performance comparison across four model configurations. Each model was trained five times with different random initializations to assess stability.

As shown in Figure 2, the proposed ResNet+FPN+Transformer architecture achieves 80.90% accuracy, demonstrating substantial improvements over the baseline ResNet50 (77.69%). Both the FPN and Transformer components provide significant contributions: FPN adds 2.35 percentage points through multi-scale feature fusion, while the Transformer contributes 2.77 points through global context modeling.

Component Comparison. Direct comparison between ResNet+FPN (80.04%) and ResNet+Transformer (80.46%) reveals that the Transformer provides a slightly higher individual improvement (+0.42%). This suggests that while both components are valuable, global context modeling has a marginally stronger impact than multi-scale features for overall accuracy. However, the similar magnitude of improvements (2.35% vs 2.77%) indicates that both components address important but complementary aspects of the problem.

Synergistic Effect. Our full hybrid architecture achieves 3.21 percentage points improvement over the baseline. While the combined improvement is less than the theoretical sum of individual components (2.35% + 2.77% = 5.12%),

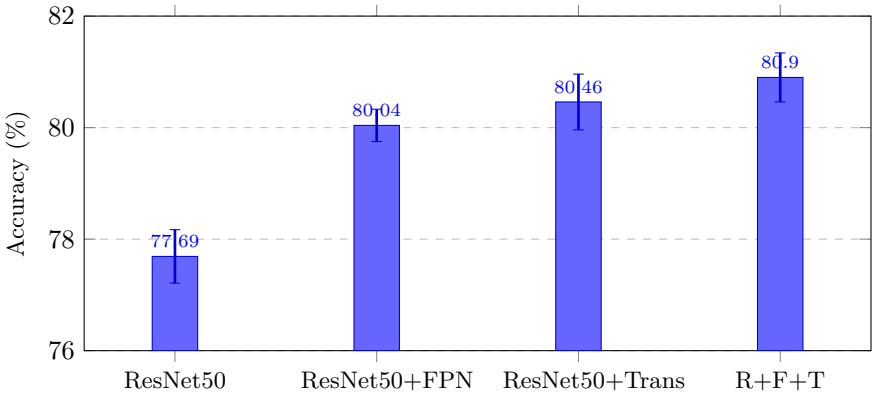


Figure 2. Ablation study results on FER-2013 dataset. Mean accuracy over 5 independent runs with error bars showing standard deviation ($\pm 1\sigma$).

this reflects the natural interaction between components where some features overlap. The integration successfully leverages the complementary strengths of both FPN and Transformer, as demonstrated by the superior per-class performance across all emotion categories. The low standard deviations across all configurations (0.29%–0.50%) indicate stable training across different initializations.

5.3. Per-class performance analysis

To understand how different architectural components affect recognition across emotion categories, we analyze per-class performance for all model variants in Figure 3.

Figure 3 reveals distinct performance patterns across emotion categories, demonstrating how different architectural components contribute to recognition of specific expressions.

ResNet+FPN Performance. The FPN component demonstrates balanced improvements across most categories, particularly excelling at fear (47.31%), surprise (86.04%), and neutral (90.89%). This suggests that multi-scale feature extraction effectively captures subtle facial details crucial for these expressions, such as the fine-grained texture patterns around the eyes in fear and the overall facial relaxation characteristic of neutral expressions.

ResNet+Transformer Performance. The Transformer component shows exceptional performance on angry expressions (78.15%), achieving an 8.61% improvement over the FPN-based model. This indicates that global context modeling is particularly beneficial for expressions characterized by complex spatial relationships, where the coordination between multiple facial regions (furrowed brows, tightened lips, and tensed jaw) must be jointly considered. However, the Transformer shows reduced performance on fear (38.71%, -8.60% vs FPN), suggesting

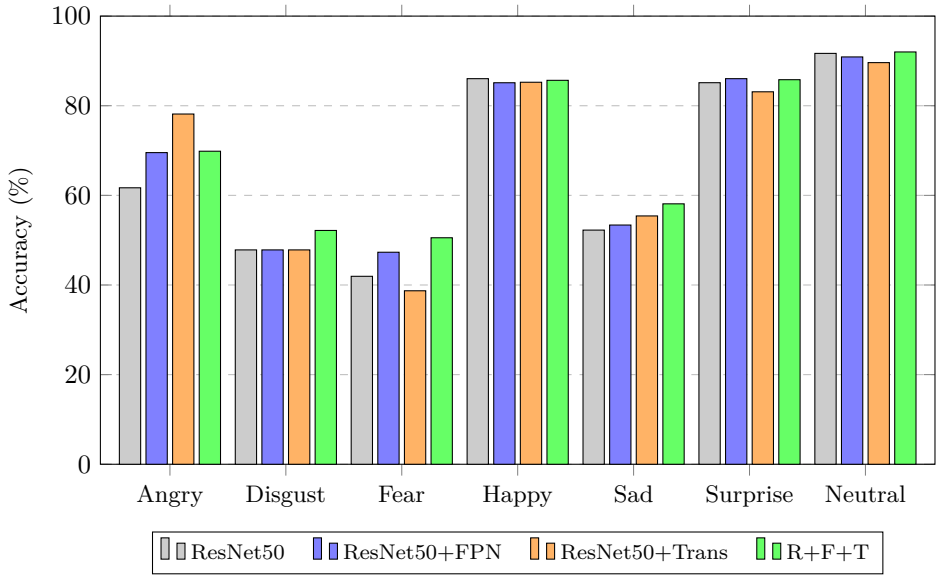


Figure 3. Per-class accuracy comparison across different architectures on FER-2013 dataset. Results averaged over 5 independent runs.

that purely global features may miss fine-grained local details critical for this emotion.

Complementary Strengths. Direct per-class comparison reveals that FPN and Transformer excel at different emotion categories. FPN outperforms Transformer on fear (+8.60%), surprise (+2.93%), and neutral (+1.27%), while Transformer excels on angry (+8.61%) and sad (+2.03%).

Full Model Advantages. Our complete ResNet+FPN+Transformer architecture successfully integrates these complementary strengths. The full model achieves best overall performance on neutral (92.00%) and fear (50.54%), demonstrates improved robustness on underrepresented classes like disgust (52.17% vs 47.83% baseline), and maintains more balanced performance across all emotion categories. Notably, while neither FPN nor Transformer alone improves disgust recognition, their combination yields a 4.34 percentage point improvement, suggesting that the integration enables the model to better handle challenging minority classes.

The confusion matrix in Figure 4 reveals the classification patterns of our model. Fear and surprise expressions show some confusion due to similar visual characteristics such as widened eyes. Sad and neutral expressions also demonstrate moderate confusion, while happy expressions show the least confusion with other categories.

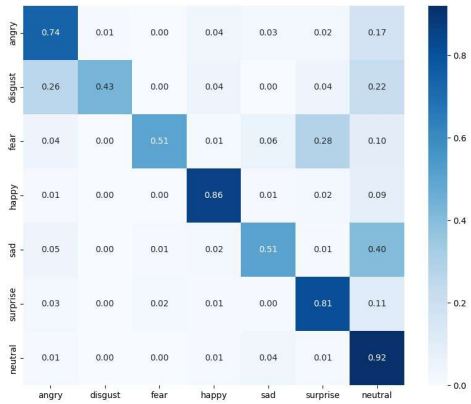


Figure 4. Confusion matrix for ResNet-FPN-Transformer model showing classification patterns.

5.4. Discussion

Dataset limitations

The FER-2013 dataset, while widely used as a benchmark, presents several inherent limitations that constrain the interpretation of our results:

Low Resolution and Grayscale: The 48×48 grayscale images limit the model's ability to capture fine-grained facial details and color-based cues that may be relevant for expression recognition in higher-quality images.

Cultural Bias: FER-2013 predominantly contains Western facial expressions, potentially limiting generalization to cross-cultural contexts where expression interpretation may differ.

Label Noise: The crowdsourced annotation process introduces label inconsistencies, as subjective interpretation of subtle expressions varies across annotators.

Performance positioning

While some recent approaches report higher accuracies on FER-2013 through specialized techniques such as ensemble methods, extensive data augmentation, or larger model architectures, our work prioritizes practical deployment considerations. Our 80.90% accuracy demonstrates effective integration of multi-scale features and global context modeling, while maintaining advantages in privacy (client-side inference), interpretability (3D visualization), and accessibility (browser-based deployment without specialized hardware).

6. Web application

We developed a web application that provides an integrated platform for facial expression recognition with interactive visualization capabilities.

6.1. Client-side architecture

Our web application implements a fully client-side architecture using TensorFlow.js, ensuring privacy by processing facial data entirely in the browser. The system comprises three main modes:

Image Analysis: Users upload images for static expression prediction with confidence visualization and feature map analysis.

Real-Time Recognition: WebRTC-based camera access enables continuous facial expression analysis with frame processing control for responsive performance.

3D Network Visualization: An interactive Three.js-based visualization allows users to explore the neural network architecture, with nodes representing different layer types and connections showing data flow.

6.2. Web deployment architecture

6.2.1. TensorFlow.js model conversion

The trained model undergoes conversion to TensorFlow.js format [5] for browser deployment. The conversion process includes weight quantization, layer optimization, and format adaptation to web-compatible tensor operations.

6.2.2. Client-side inference pipeline

The browser-based inference implements efficient preprocessing and prediction. The preprocessing pipeline includes image resizing to 48×48 pixels using bilinear interpolation, RGB to BGR channel conversion for ResNet compatibility, ResNet mean normalization ([103.939, 116.779, 123.68]), and batch dimension expansion for model input.

Memory Management: The system implements tensor disposal and parallel execution for efficient real-time processing.

Performance Optimization: Real-time processing uses parallel execution for prediction and feature map extraction through `Promise.all()`, maintaining responsive user interaction while processing facial expressions.

6.2.3. WebRTC integration

Camera access utilizes WebRTC APIs for cross-browser compatibility. The implementation includes frame rate control to balance processing load with visual responsiveness. Error handling manages camera access permissions and device compatibility issues.

6.3. Interactive visualization features

6.3.1. 3D neural network rendering

Figure 5 demonstrates the 3D network visualization interface. Users can rotate, zoom, and interact with the network structure to understand the data flow and component relationships. Different geometric shapes represent various layer types, with color coding indicating layer functions and activation levels.

The Three.js-based visualization[1] creates interactive representations of the neural network architecture:

Geometric Layer Mapping: Different layer types receive distinct visual representations:

- Convolutional layers: Cube geometries with dimensions reflecting kernel sizes
- Pooling layers: Pyramid shapes indicating dimensionality reduction
- Dense layers: Sphere geometries scaled by neuron count
- Transformer layers: Octahedron shapes distinguishing attention mechanisms

Spatial Layout Algorithm: Node positioning implements hierarchical arrangements based on network depth. The algorithm calculates appropriate spacing to maintain visual clarity while preserving logical data flow relationships.

Interactive Controls: User interaction includes mouse/touch rotation and zooming controls, click events revealing detailed layer information, and animation controls demonstrating forward pass data flow.

6.3.2. Feature map visualization system

The feature map visualization in Figure 6 provides insights into model decision-making by displaying activation patterns across network layers. The system shows how early layers capture edges and textures, while deeper layers focus on emotion-specific abstractions. The animated progression helps users understand the hierarchical feature learning process.

The feature map extraction system processes intermediate network activations. The system extracts activations from multiple network layers during inference and selects representative channels (4 channels per layer) for visualization.

[1] R. Cabello: Three.js – JavaScript 3D library, https://threejs.org/

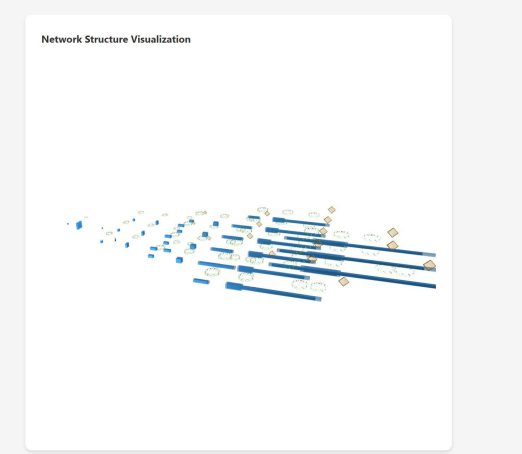


Figure 5. Interactive 3D visualization.

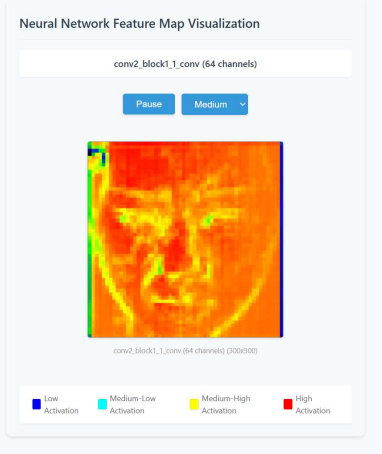


Figure 6. Feature maps.

Visualization Rendering. The system displays feature maps as animated heatmaps, showing the progression from edge detection in early layers to emotion-specific abstractions in deeper networks. Color-coded intensity maps reveal which facial regions activate different network components.

Real-time Animation. Feature map updates synchronize with inference operations, providing immediate visual feedback about network decision-making processes. The animation sequence demonstrates how facial features propagate through the network hierarchy.

6.4. Performance optimization

The complete system implements several optimization techniques:

Adaptive Rendering: Visualization quality adjusts based on device capabilities, maintaining smooth interaction across different hardware configurations.

Lazy Loading: Components initialize only when needed, reducing initial application load times and memory usage.

Efficient Resource Management: WebGL contexts and Three.js objects undergo proper cleanup to prevent resource leaks during extended usage sessions.

6.5. Ethical considerations

Privacy Protection. Our fully client-side architecture processes all facial data locally in the user's browser without server transmission, providing inherent privacy

advantages over cloud-based systems. No facial images or extracted features leave the user's device.

Fairness and Bias. Facial analysis systems may exhibit performance disparities across demographic groups. Future work should evaluate our model's fairness across age, gender, and ethnicity to ensure equitable performance.

Appropriate Use. We emphasize that emotion recognition technology should complement rather than replace human judgment, particularly in sensitive applications such as mental health assessment or surveillance contexts.

7. Conclusion

We presented a hybrid architecture combining multi-scale features and global context modeling for facial expression recognition, with client-side deployment and interactive visualizations. Future work should address cross-cultural validation and temporal modeling for video analysis.

While our system demonstrates several innovations, we acknowledge key limitations: (1) evaluation on a single, imbalanced dataset with inherent quality constraints, (2) performance gap (approximately 4-5 percentage points) compared to state-of-the-art methods, and (3) the need for further validation on diverse, real-world data to assess practical robustness across demographic groups and environmental conditions.

Key innovations include the hybrid architecture design, comprehensive training methodology addressing class imbalance, and interactive visualizations bridging the gap between technical AI implementations and human understanding. The 3D network visualization and feature map animations provide valuable educational insights into neural network behavior.

Future work includes extending to cross-cultural expression analysis, incorporating temporal information for video sequences, and developing mobile-optimized versions through model compression techniques.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby: *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, in: International Conference on Learning Representations (ICLR), Published at ICLR 2021; originally posted to arXiv in 2020, 2021, DOI: 10.48550/arXiv.2010.11929.

[2] K. He, X. Zhang, S. Ren, J. Sun: *Deep Residual Learning for Image Recognition*, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, DOI: 10.1109/CVPR.2016.90.

[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie: *Feature Pyramid Networks for Object Detection*, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 936–944, DOI: 10.1109/CVPR.2017.106.

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár: *Focal Loss for Dense Object Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence 42.2 (2020), DOI registered in 2018, published in 2020, pp. 318–327, DOI: 10.1109/TPAMI.2018.2858826.

[5] D. Smilkov, N. Thorat, Y. Assogba, A. Yuan, N. Kreeger, P. Yu, K. Zhang, S. Cai, E. Nielsen, D. Soergel, S. Bileschi, M. Terry, C. Nicholson, S. N. Gupta, S. Sirajuddin, D. Sculley, R. Monga, G. Corrado, F. B. Viégas, M. Wattenberg: *TensorFlow.js: Machine Learning for the Web and Beyond*, in: Proceedings of the 2nd SysML Conference, 2019, 2019, DOI: 10.48550/arXiv.1901.05350.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin: *Attention is All You Need*, in: Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008, DOI: 10.48550/arXiv.1706.03762.

[7] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz: *mixup: Beyond Empirical Risk Minimization*, in: International Conference on Learning Representations (ICLR), Originally posted to arXiv in 2017, 2018, DOI: 10.48550/arXiv.1710.09412.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 115–128
DOI: 10.17048/fmfai.2025.115

Detection of God Class and Data Class code smells based on an automatic machine learning tool

Nasraldeen Alnor Adam Khleel, Károly Nehéz

Department of Information Engineering, University of Miskolc,
Miskolc, H-3515, Hungary
{nasr.alnor,aitnehez}@uni-miskolc.hu

Abstract. Code smells are symptoms of poor design or incomplete implementation that can degrade software quality and maintainability. Detecting them is crucial for improving software reliability and guiding refactoring efforts.

Traditional detection methods rely on predefined rules or thresholds, which are inflexible and prone to errors, while modern machine learning approaches require significant expertise and large, balanced datasets.

To address these challenges, we propose an automated code smell detection method using AutoGluon, an AutoML framework that streamlines model selection, hyperparameter tuning, and handling of imbalanced datasets.

To evaluate the effectiveness of the proposed method, experiments were conducted using two code smell datasets: God Class and Data Class. The performance of the method was evaluated using six different metrics: accuracy, precision, recall, F-measure, Matthew's correlation coefficient (MCC), and the area under the receiver operating characteristic curve (AUC).

Additionally, we have also compared our proposed method with state-of-the-art code smell detection methods. Experimental results show that AutoGluon achieves high predictive performance—up to 0.98 accuracy for God Class and 1.00 for Data Class, which often matches or outperforms state-of-the-art methods, demonstrating the potential of AutoGluon for efficient and scalable code smell detection.

Keywords: code smells, software metrics, machine learning, AutoGluon Tool

1. Introduction

Code smells are signs of poor design that go against basic design rules [10, 12]. Finding these issues is important because it helps fix and improve the code, making

the software better and less likely to fail. These problems usually happen when developers are in a rush, use weak designs, or write quick but flawed code [10].

Software metrics play a crucial role in detecting code smells by providing measurable data about the quality and structure of code. These metrics act as objective indicators that help developers assess the health of a software system, allowing them to pinpoint areas that may require refactoring [3, 12, 13]. By analyzing different aspects such as complexity, coupling, cohesion, and size, software metrics help identify potential design flaws that may lead to maintainability issues, poor performance, or increased technical debt. When left unaddressed, these issues can make the codebase harder to understand, modify, and scale, ultimately increasing development costs and the risk of software failures [7, 16].

To enhance the accuracy and efficiency of code smell detection, machine learning techniques can be applied to analyze software metrics and automatically classify code as clean or smelly [4, 5]. Supervised learning algorithms, such as decision trees, random forests, and deep learning models, can be trained on historical datasets containing labeled code samples with identified code smells. These models learn patterns from the software metrics and can predict the presence of code smells in new, unseen code [1, 11].

Traditional methods for detecting code smells rely on rigid rules and static thresholds, which lack adaptability to different projects, require high maintenance, and often ignore the broader context of the code, leading to inaccuracies. These methods also struggle to scale for large or complex software systems. Modern machine learning approaches, while more adaptable, face challenges such as dependency on large labeled datasets, difficulty handling imbalanced data, the need for expert knowledge to tune models, high computational costs, and the risk of overfitting [12, 13, 16].

One powerful tool for automating the detection of code smells using machine learning is AutoGluon. AutoGluon is an open-source AutoML framework that simplifies the process of training and tuning machine learning models. AutoML frameworks provide a helpful solution for both beginners and experts in machine learning. For beginners, they make it easier to build high-performing ML models by handling complex tasks automatically. For experts, AutoML allows them to set up best practices—like choosing models, combining multiple models, tuning settings, preparing data, and splitting datasets—just once. After that, they can apply these steps repeatedly without needing to do everything manually. This helps experts use their knowledge more efficiently across different projects without constant hands-on work.

So, AutoGluon can address the challenges of traditional and modern techniques in detecting code smells by automating model selection, hyperparameter tuning, and handling imbalanced data with built-in techniques like class weighting and resampling, making code smell detection more efficient, scalable, and accessible without requiring extensive expertise or manual adjustments. By using AutoGluon, developers can easily apply machine learning to detect code smells without requiring deep expertise in model selection and hyperparameter tuning.

By leveraging software metrics with machine learning techniques, particularly with AutoGluon, developers can build intelligent, automated systems for detecting code smells[6, 17].

The contributions of this research can be summarized as follows: (i) Development of an automated code smell detection methodology using AutoGluon (AutoML): This study introduces a novel approach that automates model selection, tuning, and evaluation for code smell detection, reducing manual intervention and improving efficiency in software quality assessment. (ii) Comprehensive empirical evaluation on real-world datasets: Various AutoGluon models were evaluated on real software datasets using multiple performance metrics, addressing challenges such as class imbalance and identifying the most impactful software metrics through feature importance analysis. (iii) Facilitation of scalable and reproducible code quality assessments: The research contributes a practical and data-driven methodology that can be integrated into software development workflows, providing consistent, automated, and interpretable code smell detection, thus supporting empirical software engineering research and industrial applications.

2. Related work

Many traditional and modern methods for detecting code smells have been proposed in previous research works [1, 3–5, 11–13, 16].

Arcelli et al. [3] presented an approach for identifying code smells that involves the use of various ML techniques. The results indicate that all techniques performed satisfactorily; however, the imbalanced data adversely affected the performance of certain models.

Mhawish and Gupta [16] presented an approach for predicting code smells using ML techniques and software metrics. The authors utilized datasets obtained from Fontana et al., and their experimental results showed that the accurate prediction of code smells can be significantly facilitated by employing ML techniques.

Cruz et al. [4] conducted an assessment of seven ML algorithms to identify four distinct types of code smells, while also analyzing the influence of software metrics on the detection of code smells. The experimental results found that ML algorithms can perform well in detecting bad code smells, and metrics play a fundamental role in detecting bad code smells.

Dewangan et al. [5] proposed an approach based on six ML algorithms to predict code smells based on four datasets obtained from 74 open-source systems. The proposed approach's effectiveness was assessed using various performance metrics, and two feature selection methods were implemented to improve the accuracy of the predictions. The experimental results showed that their approach achieved high prediction accuracy.

Khleel and Nehéz [1, 11–13] presented various classical machine and advanced learning algorithms with different data balancing methods to detect code smells based on a set of Java projects. The authors examined four datasets related to code smells (God class, data class, feature envy, and long method) and compared

the results using various performance metrics. The experiments demonstrated that the models proposed, along with data balancing methods, exhibited improved performance in detecting code smells. In addition, the results were compared with those of state-of-the-art code smell detection methods. A comparison of the experimental results indicates that their method outperforms state-of-the-art code smell detection methods.

After reviewing previous studies in code smells detection, based on our knowledge, there are no studies that applied AutoML tools for this issue. Therefore, our study focuses on applying a new method for code smell detection, which is based on the AutoGluon Tool.

3. Proposed methodology

In this study, we present a systematic approach for automated code smell detection using AutoGluon. The choice of methodology in this study was guided by the need for scalability, automation, and robustness in code smell detection. So, AutoGluon was selected as the AutoML framework due to its ability to automate feature selection, model tuning, and ensemble construction while effectively handling imbalanced datasets [6], compared with traditional workflows such as Decision Trees, Random Forests, Support Vector Machines, k-Nearest Neighbors, and XGBoost, as well as deep learning models including CNN, LSTM, and GRU that rely on manual thresholding or tuning [3, 12, 13, 16].

Experimental results demonstrate that AutoGluon-based models achieve high accuracy—up to 0.98 for God Class and 1.00 for Data Class, which often outperform or match the state-of-the-art, highlighting the potential of AutoML to deliver accurate, efficient, and reproducible code smell detection suitable for integration into continuous quality assurance processes.

The methodology follows a structured pipeline that includes key stages such as software metrics, data modeling and collection, data preprocessing, feature selection, models building and performance evaluation. Each of these steps plays a crucial role in ensuring the accuracy and effectiveness of the detection model. Figure 1 provides an overview of the proposed methodology, with detailed explanations of each stage outlined in the following sections.

3.1. Software metrics, data modeling and collection

Software metrics are crucial for creating prediction models that help improve software quality by identifying and predicting software defects, such as bugs and code smells [12]. These metrics reveal patterns and signs that are linked to issues in the software [13]. Many studies have shown that these metrics are effective in predicting vulnerabilities in the code [12]. These metrics reveal patterns and signs that are linked to issues in the software [3, 11–13]. Additionally, researchers have demonstrated that software metrics can also be used to evaluate how reusable a piece of software is [14, 19].

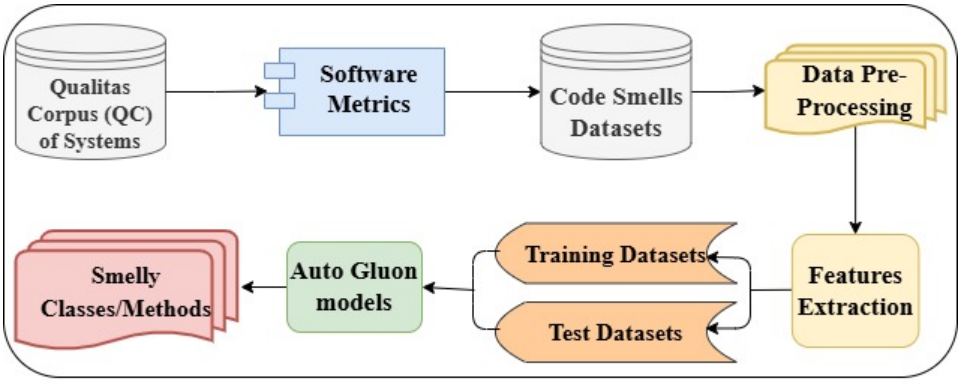


Figure 1. Overview of the Proposed Method for Code Smells Detection.

There are two main types of software metrics: static code metrics and process metrics. Static code metrics are directly extracted from the source code, while process metrics come from the source code management system, based on historical changes in the code over time. Process metrics reflect how the code evolves, including changes in the code itself, the number of changes made, and information about the developers [1, 11].

In various studies, McCabe's Cyclomatic Complexity and Halstead metrics were commonly used as independent variables to analyze code smells. McCabe's Cyclomatic Complexity measures the number of independent control paths in a program, indicating its structural complexity [13]. So, McCabe's Cyclomatic Complexity metrics focus on software quality, including cyclomatic complexity, essential complexity, design complexity, and lines of code [13].

Halstead metrics calculate program length, volume, difficulty, and effort based on operators and operands, reflecting the cognitive complexity of the code. Halstead divides software metrics into three categories: base measures, derived measures, and lines of code [3, 13, 15].

Choosing the right dataset is a key step in machine learning (ML) because classification models work better when the dataset closely matches the problem being studied. In this study, the detecting code smells models use supervised learning, which depends on a large set of software metrics as input data. Having well-structured datasets is important for training ML models effectively and ensuring that the results can be applied to different cases [10, 13].

For our analysis, we used the Qualitas Corpus, a collection of software systems compiled by researchers E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble [20]. This dataset includes many Java-based systems of different sizes and application types, as listed in Table 1, and Table 2 Lists the two specific code smells that we have investigated.

Table 1. Summary of project characteristics [3].

Number of systems	Lines of code	Number of packages	Number of classes
74	6,785,568	3420	51,826

Table 2. Lists the two specific code smells that we have investigated [3].

Code smells	Description	Affected entity
God Class	A God Class is a type of code smell that occurs when a single class takes on too many responsibilities, violating the Single Responsibility Principle.	Class
Data Class	A Data Class is a type of code smell where a class primarily exists to store data without meaningful behavior or logic.	Class

3.2. Data pre-processing and feature selection

Pre-processing the collected data is one of the critical stages before constructing the model. To generate a high-performing model, data quality must be taken into account [12]. Not all data collected is immediately suitable for training and model development. The quality of input features has a significant impact on model performance and, ultimately, prediction outcomes.

Data pre-processing encompasses a series of techniques aimed at improving the dataset by handling noise, removing irrelevant outliers, addressing missing values, and converting feature types to compatible formats. In this study, a clean and validated dataset was used to minimize the need for extensive preprocessing [12, 16].

Normalization was applied to scale numerical feature values to a uniform range (0 to 1), which helps enhance model learning efficiency [11, 16]. Specifically, Min–Max normalization was used. The normalization formula used is described in Equation (3.1).

Feature Selection is another crucial component of the modeling pipeline, as it aims to reduce dimensionality by retaining only the most informative and discriminative features relevant to the target variable [13]. This process eliminates irrelevant, redundant, or noisy variables, which may otherwise decrease model accuracy and increase training time [1, 4].

In this study, we adopted an embedded feature selection method, which is inherently integrated within the model training process. Unlike filter methods that evaluate features independently of any learning model, or wrapper methods that evaluate subsets of features through repetitive training cycles (which can be computationally expensive), embedded methods assess feature importance during model construction. This allows the algorithm to automatically prioritize features that improve performance and ignore those that do not contribute meaningfully.

AutoGluon, the AutoML framework used in this study, performs this embedded

selection as part of its training pipeline, making it both computationally efficient and well-suited for handling high-dimensional software metric datasets. Additionally, feature scaling was applied to ensure that all selected features were on a comparable scale, further supporting consistent learning behavior across models.

$$x_i = (x_i - Xmin)/(Xmax - Xmin) \tag{3.1}$$

Where max(x) and min(x) represent the maximum and minimum value of the attribute x, respectively.

3.3. Models building and evaluation

In this study, model development and evaluation were conducted using AutoGluon, an open-source AutoML framework that automates the machine learning pipeline and supports both novice and advanced users. AutoGluon simplifies tasks such as data preprocessing, feature engineering, model selection, and hyperparameter tuning, enabling rapid development of high-performing models.

It supports a diverse range of algorithms, including LightGBM, RandomForestEntr, LightGBMXT, XGBoost, and deep neural networks, all of which are automatically trained and combined through ensemble techniques such as WeightedEnsemble-L2 [2, 8]. For this work, the dataset was split into 80% for training and validation (handled internally through cross-validation) and 20% for independent testing. AutoGluon applied classification-appropriate defaults such as log loss for optimization and automated selection of learning rates and batch sizes.

The evaluation of the trained models was based on standard metrics derived from the confusion matrix, including accuracy, precision, recall, F1-score, and MCC, which is a statistical metric used to assess the performance of binary classification models. It takes into account true and false positives and negatives and is regarded as a balanced measure, even if the classes are of very different sizes. Additionally, AUC was employed to assess the discriminative ability of the classifiers, illustrating the trade-off between True Positive (TP) and False Positive (FP) rates across different thresholds [9, 18].

As shown in Figure 2, the confusion matrices for each dataset confirm the models' effectiveness in predicting smelly and non-smelly code. The mathematical formulations of these metrics are defined in Equations (3.2) to (3.7).

Overall, AutoGluon's automation and ensemble strategy proved effective for detecting code smells such as God Class and Data Class, delivering robust performance with minimal manual intervention.

$$Accuracy = ((TP + TN))/((TP + FP + FN + TN)) \tag{3.2}$$

$$Precision = (TP)/((TP + FP)) \tag{3.3}$$

$$Recall = (TP)/((TP + FN)) \tag{3.4}$$

$$F - Measure = ((2 * Recall * Precision))/((Recall + Precision)) \tag{3.5}$$

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.6}$$

$$\text{AUC} = \frac{\sum_{\text{ins}_i \in \text{Positive Class}} \text{rank}(\text{ins}_i) - \frac{M(M+1)}{2}}{M \cdot N} \tag{3.7}$$

4. Experimental results and discussion

The experimental evaluation aimed to assess the performance of the proposed AutoGluon-based methodology for detecting code smells using real-world datasets, which are God Class and Data Class, sourced from the Qualitas Corpus. Multiple models were automatically trained and optimized by AutoGluon, including LightGBM, RandomForestEntr, LightGBMXT, XGBoost, and the ensemble model WeightedEnsemble-L2. Each model's performance was evaluated using standard classification metrics such as accuracy, precision, recall, F1-score, MCC, and AUC. As shown in Tables 3 and 4, for the God Class dataset, all five top-performing models (including LightGBM, XGBoost, and LightGBMXT) achieved high levels of accuracy, ranging from 0.97 to 0.98, with corresponding precision and recall values consistently reaching 0.96 or higher. The MCC and AUC values for these models also remained close to or at 1.00, indicating excellent discriminatory power. Similarly, for the Data Class dataset, the models demonstrated outstanding performance, with LightGBM, RandomForestEntr, and WeightedEnsemble-L2 achieving perfect scores (1.00) across all metrics, while XGBoost and LightGBMXT followed closely with slightly lower, yet still impressive, values.

Table 3. Evaluation results for the top five models – God Class dataset.

Models	Performance Measures					
	Accuracy	Precision	Recall	F-measure	MCC	AUC
LightGBM	0.97	0.94	1.00	0.96	0.95	1.00
RandomForestEntr	0.98	0.96	1.00	0.98	0.97	1.00
LightGBMXT	0.98	0.96	1.00	0.98	0.97	0.99
XGBoost	0.98	0.96	1.00	0.98	0.97	0.98
WeightedEnsemble-L2	0.98	0.96	1.00	0.98	0.97	0.99

In terms of training efficiency (Table 5), the AutoGluon-based models achieved consistently strong performance on the God Class dataset. The best results were achieved by LightGBMXT and WeightedEnsemble-L2, both reaching a validation accuracy of 0.985 and a test accuracy of 0.988. These models also had relatively low

Table 4. Evaluation results for the top five models – Data Class dataset.

Models	Performance Measures					
	Accuracy	Precision	Recall	F-measure	MCC	AUC
LightGBM	1.00	1.00	1.00	1.00	1.00	1.00
RandomForestEntr	1.00	1.00	1.00	1.00	1.00	1.00
LightGBMXT	0.98	1.00	0.97	0.98	0.97	0.99
XGBoost	0.97	1.00	0.94	0.97	0.95	1.00
WeightedEnsemble-L2	1.00	1.00	1.00	1.00	1.00	1.00

training times (0.36s and 0.45s, respectively), showing a strong balance between accuracy and efficiency. Notably, XGBoost performed similarly in test accuracy (0.988) but trained faster (0.175s), making it the most efficient model in terms of runtime. For the Data Class dataset, performance was even more impressive. LightGBM and WeightedEnsemble-L2 achieved perfect scores on both validation and test sets (1.000). While LightGBMXT required significantly longer training time (2.48s), it still maintained high accuracy (0.985 validation, 0.988 test).

Table 5. Training Time (seconds) and Models Performance.

Models	God Class Dataset		
	fit-time	score-val	score-test
LightGBM	0.405785	0.970588	0.976190
RandomForestEntr	0.657616	0.970588	0.988095
LightGBMXT	0.361412	0.985294	0.988095
XGBoost	0.175200	0.970588	0.988095
WeightedEnsemble-L2	0.450221	0.985294	0.988095
Models	Data Class Dataset		
	fit-time	score-val	score-test
LightGBM	0.273476	1.000000	1.000000
RandomForestEntr	0.646896	0.985294	1.000000
LightGBMXT	2.481219	0.985294	0.988095
XGBoost	0.180894	0.985294	0.976190
WeightedEnsemble-L2	0.360617	1.000000	1.000000

The confusion matrices (Figure 2) and AUC (Figure 3) visually confirmed the models' ability to accurately distinguish between smelly and non-smelly classes. The confusion matrices underline the efficacy of AutoGluon models in accurately detecting code smells. The minimal or absent false classifications demonstrate the robustness of these models. In addition to accuracy, precision, recall, F-measure, and MCC, we also report AUC scores to assess the discriminative ability of the classifiers. As shown in Tables 3 and 4, AUC values for both God Class and Data Class datasets are consistently high, ranging from 0.98 to 1.00 across all models. These

results confirm that the classifiers are not only highly accurate but also robust in distinguishing smelly from non-smelly classes. This finding is particularly important when dealing with imbalanced datasets, where accuracy alone can sometimes mask poor performance in minority classes. The consistently high AUC values demonstrate that AutoGluon-based models achieve excellent sensitivity-specificity trade-offs, reinforcing their suitability for automated software quality assurance tasks.

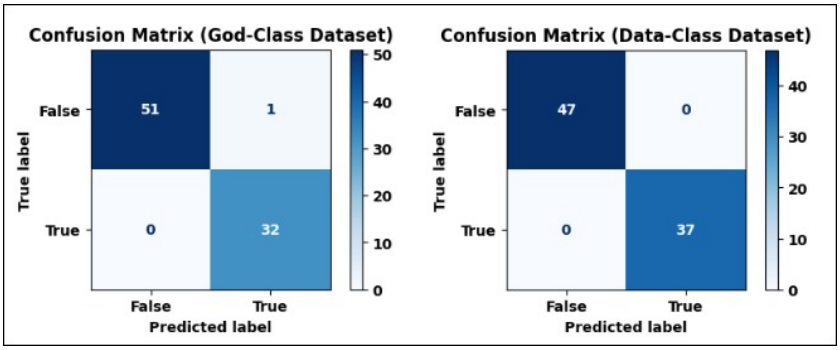


Figure 2. Confusion Matrix for the models over all datasets.

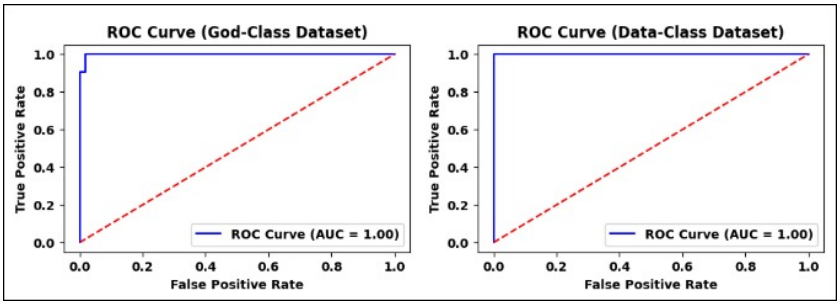


Figure 3. AUC for the models over all datasets.

AutoGluon provides feature importance scores based on permutation shuffling, which quantifies how much the model's performance decreases when each feature is randomly shuffled. Higher importance scores indicate that the model relies more on that feature for making predictions. Therefore, feature importance analysis (Figures 4 and 5) revealed that a subset of software metrics significantly influenced the prediction outcomes, validating the effectiveness of embedded feature selection techniques used in the pipeline.

In comparison with previous state-of-the-art approaches (Table 6). We compared our results with the results obtained in previous studies based on the accuracy. The values marked with "-" indicate that the approaches that did not use data balancing techniques or did not provide results for the performance measure

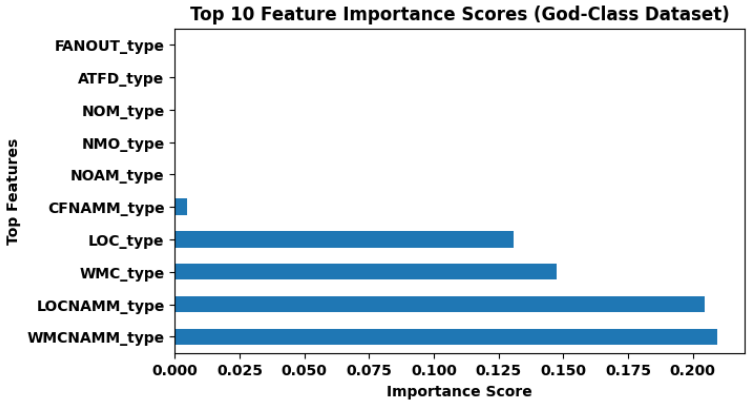


Figure 4. Feature importance scores for the models – God Class Dataset.

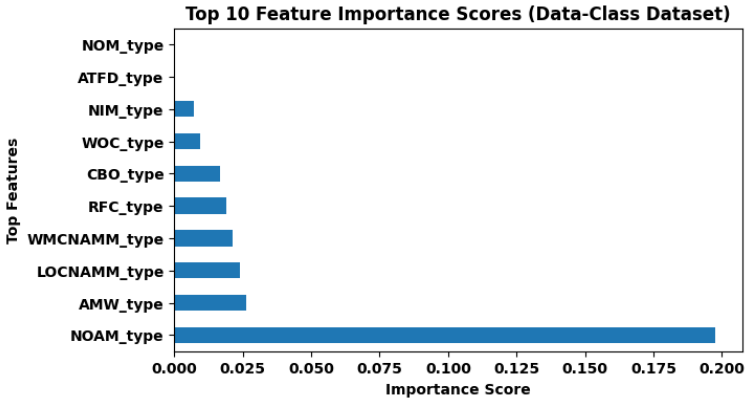


Figure 5. Feature importance scores for the models – Data Class Dataset.

in a particular data set. Additionally, our proposed models are highlighted in bold text. According to the Table, some of the results in the previous studies are better than ours, but in most cases, our method outperforms the other state-of-the-art approaches and provides better predictive performance. These findings collectively demonstrate the practicality, robustness, and accuracy of the AutoGluon framework in automating code smell detection, reducing reliance on human expertise, and offering scalable solutions for software quality assurance. Future studies could focus on testing these models on larger datasets or in real-world scenarios to further validate their effectiveness and generalizability.

Temporal and Long-Term Development Context of God and Data Classes: Code smells such as God Classes and Data Classes rarely emerge instantaneously; in-

Table 6. Comparison of the proposed models with other existing approaches based on accuracy.

Approaches	**Data Balancing Techniques**	**God Class**	**Data Class**
Decision Tree [12]	Random oversampling	0.98	1.00
K-Nearest Neighbors [12]	Random oversampling	0.97	0.96
Support Vector Machine [12]	Random oversampling	0.96	0.97
XGBoost [12]	Random oversampling	0.96	1.00
Multi-Layer Perceptron [12]	Random oversampling	0.97	0.98
Bi-LSTM [13]	Random oversampling	0.96	0.99
GRU [13]	Random oversampling	0.96	0.98
Bi-LSTM [13]	Tomek links	0.96	0.95
GRU [13]	Tomek links	0.96	0.99
Random Forest [3]	–	0.96	0.98
Naive Bayes [3]	–	0.97	0.97
Random Forest [16]	–	–	0.99
CNN [6]	SMOTE	0.96	0.98
XGBoost [2]	SMOTE	0.99	–
SVM [2]	SMOTE	0.97	–
KNN [2]	SMOTE	0.97	–
Random Forest [8]	–	0.69	0.70
Naive Bayes [8]	–	0.82	0.75
SVM [8]	–	0.74	0.83
KNN [8]	–	0.80	0.82
Our proposed LightGBM	–	0.97	1.00
Our proposed RandomForestEntr	–	0.98	1.00
Our proposed LightGBMXT	–	0.98	0.98
Our proposed XGBoost	–	0.98	0.97
Our proposed WeightedEnsemble-L2	–	0.98	1.00

stead, they evolve gradually as projects grow in size and complexity, often persisting across multiple releases. Their longevity reflects not only design flaws but also the developmental pressures and shortcuts taken during software evolution. In this study, the analysis was based on static snapshots of systems from the Qualitas Corpus, so project timelines and release histories were not explicitly captured. Nevertheless, prior research indicates that both God Classes and Data Classes often become long-lived entities, contributing to technical debt across the lifecycle of software projects. Integrating AutoML-based detection into CI/CD pipelines offers a way to address this challenge by enabling continuous monitoring of these smells, allowing teams to identify their emergence early, track their growth, and guide timely refactoring. In this way, automated detection not only classifies existing design problems but also supports sustainable software evolution by mitigating the accumulation of long-lived technical debt.

5. Conclusion

Code smell detection involves identifying patterns in the source code that indicate potential problems with design or implementation, even if the code is working correctly.

This study presented an automated approach to detecting code smells using AutoGluon, an AutoML framework that simplifies the machine learning workflow by automating tasks such as model selection, hyperparameter tuning, feature selection, and data balancing.

By leveraging software metrics and structured data from the Qualitas Corpus, the proposed methodology was evaluated on two prominent code smell types: God Class and Data Class. The experimental results confirmed the effectiveness of the approach, with all models achieving high accuracy, precision, recall, F1-score, MCC, and AUC.

The use of AutoGluon allowed for efficient training and performance optimization without manual tuning, making the methodology accessible to both novice and expert users. The models also showed resilience to imbalanced data and highlighted the impact of key software metrics through feature importance analysis.

Moreover, comparisons with existing state-of-the-art approaches demonstrated that the proposed method either outperformed or matched traditional and deep learning-based techniques, further validating its competitiveness and reliability.

Overall, the integration of AutoML techniques into code smell detection offers a promising pathway toward automated, interpretable, and scalable software quality assessment. Future work may include expanding the methodology to detect additional types of code smells, applying the approach to larger and more diverse software projects, and exploring hybrid AutoML strategies for further performance gains.

Acknowledgements. The authors gratefully acknowledge the financial assistance from the Institute of Information Science, Faculty of Mechanical Engineering and Informatics, University of Miskolc.

References

[1] N. A. Adam Khleel, K. Nehéz: *Optimizing LSTM for Code Smell Detection: The Role of Data Balancing.* Infocommunications Journal 16.3 (2024), DOI: 10.36244/ICJ.2024.3.5.

[2] K. Alkharabsheh, S. Alawadi, V. R. Kebande, Y. Crespo, M. Fernández-Delgado, J. A. Taboada: *A comparison of machine learning algorithms on design smell detection using balanced and imbalanced dataset: A study of God class*, Information and Software Technology 143 (2022), p. 106736, DOI: 10.1016/j.infsof.2021.106736.

[3] F. Arcelli Fontana, M. V. Mäntylä, M. Zanoni, A. Marino: *Comparing and experimenting machine learning techniques for code smell detection*, Empirical Software Engineering 21.3 (2016), pp. 1143–1191, DOI: 10.1007/s10664-015-9378-4.

[4] D. Cruz, A. Santana, E. Figueiredo: *Detecting bad smells with machine learning algorithms: an empirical study*, in: Proceedings of the 3rd international conference on technical debt, 2020, pp. 31–40, DOI: 10.1145/3387906.3388618.

[5] S. Dewangan, R. S. Rao, A. Mishra, M. Gupta: *A novel approach for code smell detection: an empirical study*, IEEE Access 9 (2021), pp. 162869–162883, DOI: 10.1109/ACCESS.2021.3133810.

[6] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola: *Autogluon-tabular: Robust and accurate automl for structured data*, arXiv preprint arXiv:2003.06505 (2020), DOI: arXivpreprintarXiv:2003.06505.

[7] T. Guggulothu, S. A. Moiz: *Code smell detection using multi-label classification approach*, Software Quality Journal 28.3 (2020), pp. 1063–1086, DOI: 10.1007/s11219-020-09498-y.

[8] S. Jain, A. Saha: *Rank-based univariate feature selection methods on machine learning classifiers for code smell detection. Evol. Intell. 15 (1), 609–638*, 2022, DOI: 10.1007/s12065-020-00536-z.

[9] S. Jain, A. Saha: *Improving performance with hybrid feature selection and ensemble machine learning techniques for code smell detection*, Science of Computer Programming 212 (2021), p. 102713, DOI: 10.1016/j.scico.2021.102713.

[10] A. Kaur, S. Jain, S. Goel, G. Dhiman: *A review on machine-learning based code smell detection techniques in object-oriented software system (s)*, Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering) 14.3 (2021), pp. 290–303, DOI: 10.2174/2352096513999200922125839.

[11] N. A. A. Khleel, K. Nehéz: *Deep convolutional neural network model for bad code smells detection based on oversampling method*, Indonesian Journal of Electrical Engineering and Computer Science 26.3 (2022), pp. 1725–1735, DOI: 10.11591/ijeecs.v26.i3.pp1725-1735.

[12] N. A. A. Khleel, K. Nehéz: *Detection of code smells using machine learning techniques combined with data-balancing methods*, International Journal of Advances in Intelligent Informatics 9.3 (2023), pp. 402–417, DOI: 10.26555/ijain.v9i3.981.

[13] N. A. A. Khleel, K. Nehéz: *Improving accuracy of code smells detection using machine learning with data balancing techniques*, The Journal of Supercomputing 80.14 (2024), pp. 21048–21093, DOI: 10.1007/s11227-024-06265-9.

[14] N. Medeiros, N. Ivaki, P. Costa, M. Vieira: *Vulnerable code detection using software metrics and machine learning*, IEEE Access 8 (2020), pp. 219174–219198, DOI: 10.1109/ACCESS.2020.3041181.

[15] B. Mehboob, C. Y. Chong, S. P. Lee, J. M. Y. Lim: *Reusability affecting factors and software metrics for reusability: A systematic literature review*, Software: Practice and Experience 51.6 (2021), pp. 1416–1458, DOI: 10.1002/spe.2961.

[16] M. Y. Mhawish, M. Gupta: *Predicting code smells and analysis of predictions: using machine learning techniques and software metrics*, Journal of Computer Science and Technology 35.6 (2020), pp. 1428–1445, DOI: 10.1007/s11390-020-0323-7.

[17] L. M. Paladino, A. Hughes, A. Perera, O. Topsakal, T. C. Akinci: *Evaluating the performance of automated machine learning (AutoML) tools for heart disease diagnosis and prediction*, Ai 4.4 (2023), pp. 1036–1058, DOI: 10.3390/ai4040053.

[18] F. Pecorelli, D. Di Nucci, C. De Roover, A. De Lucia: *A large empirical assessment of the role of data balancing in machine-learning-based code smell detection*, Journal of Systems and Software 169 (2020), p. 110693, DOI: 10.1016/j.jss.2020.110693.

[19] K. Z. Sultana, V. Anu, T.-Y. Chong: *Using software metrics for predicting vulnerable classes and methods in Java projects: A machine learning approach*, Journal of Software: Evolution and Process 33.3 (2021), e2303, DOI: 10.1002/smr.2303.

[20] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, J. Noble: *The qualitas corpus: A curated collection of java code for empirical studies*, in: 2010 Asia pacific software engineering conference, 2010, pp. 336–345, DOI: 10.1109/APSEC.2010.46.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 129–139
DOI: 10.17048/fmfai.2025.129

Efficiency testing of openset learning methods in image classification

László Kovács, Enikő Palencsár, Péter Bán

University of Miskolc, Institute of Computer Science
{laszlo.kovacs,palencsar.eniko}@student.uni-miskolc.hu
bn.peter.hun@gmail.com

Abstract. The problem of detecting untrained categories may cause efficiency degradation in many application areas, because the real-word domains are usually dynamic, or the available data set may be incomplete. Despite the relatively high cost of related misclassification errors, the field of openset learning is an underinvestigated domain in machine learning. The main goal of this paper is to investigate the efficiency of current technologies for the openset learning problem on a standard benchmark image dataset. As the results of the performed comparison tests show that the widely proposed standard methods do not provide good results, in many cases the hybrid methods can dominate the usual approaches. In the paper, we present a novel extended threshold method that provides better accuracies than the usual benchmark methods.

Keywords: image classification, CNN neural networks, openset learning problem

AMS Subject Classification: 68T07

1. Introduction

Neural networks are now the dominant technologies in complex classification and regression tasks. The neural network as a universal approximator applies a complex network of elementary functions to predict the function values at arbitrary positions. According to the General Approximation Theorem, a feedforward neural network with a single hidden layer containing a finite number of neurons and using a continuous activation function increasing monotonically can approximate any continuous function [6]. The model construction process to adjust the weight values of the neural network is optimized with a training process. The usual backprop-

agation optimization method adapts the weight values to the available supervised training dataset.

In the case of classification problems, the neurons in the output layer correspond to the different categories in the dataset and the neuron with the highest output value determines the winner category [1]. For example, in the case of image classification, the set of image categories is fixed and the training set should cover all categories in a uniform way. The problem of unbalanced class distribution is a widely investigated problem [7] as it can cause efficiency degeneration due to difficulty learning the limits of the decision or to misleading performance metrics. Thus, one of the main goals of the data preparation phase is to build a well-balanced training data set for the predefined categories.

Usually, it is hard work to meet this kind of requirement, or sometimes it is an impossible task. Our investigation focuses on the domain of related open-set learning problems [2]. Unlike the traditional situation, the test set may contain cases that do not belong to any of the categories presented in the training set. The test data set may contain instances of previously unseen classes. The key challenge here is to detect these unseen cases; the neural network should recognize that the input differs significantly from any trained categories.

The problem of detecting untrained categories may occur in many areas of application, because the real word domains are usually dynamic, or the available data set may be incomplete [5]. We can highlight the following application domains where the risk of incomplete training set is relatively high:

- image object classification, where the image contains un-trained objects widely used technology in medical diagnosis or autonomous driving,
- intruder detection,
- fault detection in industrial monitoring,
- sentiment analysis.

Despite the relatively high cost of related misclassification errors, the field of openset learning is an underinvestigated domain in machine learning. The main goal of this paper is to investigate the efficiency of current technologies for the openset learning problem on a standard benchmark image dataset.

The results of the performed comparison tests show that the widely proposed standard methods do not provide good results, in many cases the hybrid methods can dominate the usual approaches. In the paper, we also present a novel extended threshold method that provides accuracies that are better than the usual benchmark methods.

2. Related methods

The problem domain of openset learning is similar in many aspects to some other problem domains related to incomplete training datasets. One of these fields is

the one-class classification problem [15]. In the case of one-class classification, the training set contains only a single class (only positive cases), and the main goal of the prediction is to determine whether the test object belongs to this class or not [5].

The one-shot learning domain [14] refers to the case when the training set contains only a single example for each existing class. Here, the generated model should provide an optimal generalization based on a single element per class. The training process cannot memorize the common features found in the different instances of the class, it should discover the characteristic features which can be used to distinguish the different classes.

In zero-shot learning, the generated model is not based on instances, but on some available metadata, semantic information on the different classes [16]. The main challenge in this problem domain is the efficient integration of the different multi-model metadata information items.

In the investigated openset learning task, the problem domain can be characterized by the following properties [12]:

- The applied training set does not cover all classes possibly found in the production data.
- The generated model should correctly recognize all classes found in the training set.
- The method should identify the classes not contained in the training set as an outlier or an 'unknown' class.
- No additional semantic information is provided on the classes; the model is inferred only from the available instances.

In the literature, we can find several approaches to deal with this domain of open-set learning problem. The main methods are summarized in the following table.

- Threshold-based category acceptance. We apply the usual multicategory classifier neural network built on the training set. Using the softmax activation function, the output values represent the probability distribution across the different classes. If the maximum output value is below a certain threshold (no clear winner category), the test image is assigned as an outlier [4].
- The OpenMax method is a special variant of the threshold approach. It applies a Weibull distribution to involve the probability of the 'unknown' class. [18]
- Distance-based approach where the method is based on the concept of locality. If the new item is far from any training items, the tested item can be considered an outlier, unknown class [12].

- Development of a MLP neural network for similarity regression, where the similarity shows the distance of the test image to the trained categories. If the similarity is below a given threshold, the test image belongs to the new category.
- Application of generative models, autoencoder neural networks to predict the membership similarity to the known categories. Separate autoencoder neural networks are constructed for the different categories. Taking the test image, the engine generates the autoencoder output for every class. If the maximum similarity between the input and output images is below a threshold, the image is classified as an outlier.
- Outlier exposure method, where the initial training set is extended with noisy outlier data that are labeled 'unknown' class. Although this method improves the robustness of the model, noisy extension usually does not cover all possible external cases [17].

The most widely used approach for the openset problem is the OpenMax method introduced in [2] from 2016. The method first takes the activation outputs of all known classes for all input in the training set. Then it calculates the mean activation vector for each class. In the next step, the distances between the mean vector and the single activation vectors are calculated, then the top k elements with largest distances are selected, and using these values a Weibull distribution is fit for the distinct classes.

$$P(d \mid \lambda_c, k_c) = 1 - \exp\left(-\left(\frac{d}{\lambda_c}\right)^{k_c}\right)$$

For an input test item, we first calculate the standard activation vector and distance values. In the next step, using the Weibull distribution weights, a revised activation vector is calculated. The formula for the Weibull weights:

$$w_{s(i)}(x) = 1 - a(i) \exp\left(-\left(\frac{d}{\lambda_{s(i)}}\right)^{k_{s(i)}}\right)$$

Next, we take an additional activation function for the unknown class. In the last step, we apply the softmax layer for this extended revised activation vector, which also contains a probability value for the unknown class.

The Isolation Forest method [3] applies a space segmentation algorithm to isolate outlier elements in the data space. The Isolation Forest architecture is built up from more random isolation trees. For each node in the isolation tree, a random dimension (attribute) is assigned, and a binary split procedure is performed. The split procedure ends if the size of the corresponding subtree is below a given threshold. Using this methodology for the entire forest, we can calculate an average depth for each item in the data set. Those items are considered outliers or unknown cases, where this average depth is a small value.

Isolation Forests were introduced by Liu, Ting, and Zhou in 2008 [10] as an unsupervised method for fast anomaly detection. The algorithm is based on two key assumptions:

- anomalies represent a minority in the dataset, and
- their feature values differ significantly from those of the majority [11].

For each node in the tree:

- Randomly select a feature $f \in \{1, \ldots, d\}$
- Randomly select a split value $s \in [\min(X_f), \max(X_f)]$
- Partition data:
  - Left child: $X_f < s$
  - Right child: $X_f \geq s$
- Recursively apply until:
  - Node has only one instance, or
  - Tree reaches **maximum depth** $\lceil \log_2(n) \rceil$

Figure 1. Algorithm of building the isolation tree.

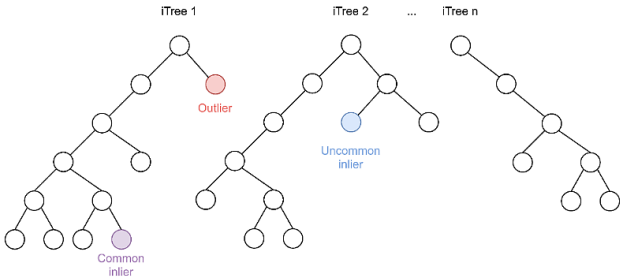


Figure 2. The structure of the isolation tree.

3. Proposed approaches

In our investigation, we propose an extended variant of the threshold-based approach. Usually, the existing methods perform a similarity check in the input space or in the output space. Our assumption is that both spaces can provide additional information about outlier items. Thus, the proposed method applies two class similarity measures for outlier detection. The first component calculates the similarity of the softmax output probability values. The method calculates the differences between the probability p_1 of the winner class and the probability p_2 of the second-best class.

$$w_1(x) = p_1(x) - p_2(x)$$

In the formula, the symbol x denotes the current element to be tested. If the model is not sure which class is the winner, this measure $w_1(c)$ is small.

The second measure is used to locate outliers in the feature space of the objects. Usually, the input space contains the required feature vectors, but in the case of images, the input matrix provides only a pixel-level description, which representation is far from the semantic-level feature vector content. To provide a better representation of semantical features, the first method version utilizes the input vectors of the last Dense layer classifier module of the CNN neural network. Thus, the output of the flattening layer is used to describe the image features.

For a given element x, we define

$$w_{2a}(x, c) = \sum_{i \in c} \exp(-d(x, x_i))$$

where the summation runs over the objects of class c. The symbol $d()$ denotes the distance between the tested item and a single object in class c. The large $w_{2a}(x, c)$ values mean that the item is near the elements of class c, and the small $w_{2a}(x, c)$ values indicate an outlier element. The proposed method calculates the weight values both for the winner class and the complement classes. In the case of outliers, where the class prediction is incorrect, there is no significant difference between the values for the winner and rejected classes.

In addition to the flattening layer method, we also applied the perceptual hashing approach. The perceptual hashing method is used to generate content-based fingerprints of images. The calculation of the description vector consists of the following steps:

- Standardization of the image, conversion to predefined size, and grayscale colormap.
- Application of a discrete cosine transform to detect the internal description using components of different frequency.
- Extraction of the most important components
- Calculation of the hash value for the selected components.

The final decision on the outlier status is calculated with the following method.

1. Generating the predicted class ($c1$) and the second-best class $c2$ for the input object x using the trained CNN neural network.
2. Calculation of $w_1(x)$.
3. If $w_1(x) < \alpha_1$ then x is an unknown outlier class.
4. Calculation of $w_2(x, c1)$.
5. If the value of w_2 is above a threshold $alpha_2$, then x is an unknown outlier class.

In addition to the extended threshold method, we have also tested an ensemble CNN version classifier, where we built up a separate classifier neural network for all classes in the data set. Each of the classifiers works as a binary classifier related to one of the classes. If the predicted class probability is below a threshold in each class, the test item will be predicted as an unknown class.

Considering the members of the ensemble, we investigated more variants of the output vector representation forms, all related to some type of layer of the CNN network. Our tests involved the following feature vector variants:

- Flattened output of convolutional layers, where a PCA reduction concept was applied to have a moderate vector size.
- Output of the fully connected layer in the MLP module.
- Logits of the output layer in the MLP unit.
- Softmax of logits.

As this architecture presented the weakest accuracy in the preliminary tests (more than 22% lower accuracy than the extended threshold method), we decided to eliminate it from the group of final candidate methods.

The third proposed variant was the application of an isolation forest architecture. This structure is a special variant of the random forest architecture, where the main goal is to efficiently locate outlier nodes. The method will partition the item into disjoint leaf nodes. If the size of the container nodes is below a threshold, the element is considered as an outlier.

In our investigation, we adapted the Isolation Tree method to the outlier detection of images, and we proposed two variants:

1. The output of the convolution module was used as the vector of attributes of the object.
2. The softmax layer output was used as the feature vector.

4. Experimental evaluation

4.1. Test environment

The main goal of the tests performed was to evaluate the proposed outlier detection methods and compare the accuracy levels achieved with a benchmark method.

In the tests, the following methods were involved:

- OpenMax algorithm as benchmark method
- Isolation forest
- Extended threshold method

The test system was implemented in Python Keras-Tensorflow framework using the Colab development environment.

For the tests, we applied the following three benchmark image classification datasets:

- CIFAR-10 [8]
- MNIST-10 [9]
- COIL-20 [13].

The CIFAR-10 benchmark dataset contains 60000 images (50000 for training and 10000 for test) of small resolution (32*32). The images belong to 10 categories. In the tests, we created reduced training sets to exclude instances of some selected categories. In the test dataset, we assign these instances to a new common category. From the point of view of category prediction in the test phase, the training dataset contains only positive examples, having only "known known" and "unknown unknown" classes [5].

The original CIFAR-10 dataset includes items from 10 classes, but we selected 3 as unknown classes, thus the CNN neural network models involved in the experiments were trained only on 7 known classes.Thus the training dataset contained 35000 images, Image resolution is (32, 32, 3). For the tests, we used all classes, the test set contained 10000 images Thus nearly 30% of the test items belonged to the unknown category.

The MNIST-10 dataset is a key benchmark in machine learning and computer vision, specifically for image classification tasks. The dataset consists of a large collection of 70,000 grayscale images of handwritten digits. The images are encoded into a pixel matrix of size (28 , 28, 1). Its primary role is to provide a standardized, simple dataset for training and testing various image classification algorithms, from classic machine learning models to complex deep learning architectures. .

The Columbia Object Image Library (COIL-20) is a well-known dataset in the field of computer vision. It consists of 1,440 grayscale images of 20 different objects. In our tests, we included only 10 categories. The images were created by placing each object on a turntable and capturing 72 images at 5-degree intervals as it was rotated through 360 degrees. This setup provides a comprehensive set of images for each object, showing it from a wide range of poses and angles.

4.2. Test results

In the efficiency tests, we measured the following accuracy values:

- T1: Validation accuracy in training of the baseline CNN model with 7 classes;
- T2: Test accuracy with 7 classes on the trained baseline CNN model;
- T3: Test accuracy with 10 classes on the trained baseline CNN model;
- T4: Test accuracy with 10 classes using the baseline OpenMax NN model;

- T5: Test accuracy with 10 classes on the proposed Extended threshold NN model;
- T6: Test accuracy with 10 classes on the proposed Isolation Forest NN model using convolution layer output;
- T7: Test accuracy with 10 classes on the proposed Isolation Forest NN model using softmax layer output.

In the tests, we performed five measurements and calculated the average and standard deviation aggregations. The resulting aggregation values are summarized in Table 1. The aggregated values are based on 5 measurements. In the first row, we see the validation accuracy at the end of the training process. We involved only 7 classes into the training, the members from 3 classes were removed from the training set. The output layer of the constructed neural network was able to recognize only 7 categories. The second row shows the accuracy of the test data set with 7 classes. In the third row, the values show the test accuracy when the test dataset contained all 10 classes. The fourth row shows the test accuracy achieved using the OpenMax method using 10 classes in the test. The fifth row relates to the Extended threshold method, while the sixth row is for the results of the Isolation tree method. All values in the table are given in percentage units.

Table 1. Results of the comparison tests.

Method measure	CIFAR avg	CIFAR stdev	MNIST avg	MNIST stdev	COIL avg	COIL stdev
T1	91.1	0.94	99.5	0.07	99.1	0.68
T2	71.5	0.54	98.8	0.41	98.7	1.56
T3	49.8	0.35	68.9	0.39	82.7	0.89
T4	48.7	0.56	69.5	0.04	82.8	0.34
T5	50.8	0.43	74.9	0.89	82.9	3.2
T6	49.0	1.94	47.8	3.12	82.3	0.24
T7	64.1	1.92	78.6	1.71	81.9	0.74

As we can see in the result table, we experienced an unexpected weak result in the case of the baseline OpenMax method. The OpenMax method achieved very similar accuracies as the baseline neural network, there was no significant improvement. We remark that we did not perform a hyperparameter tuning for the OpenMax method in our tests.

In the tests on the Extended threshold method, the proposed method consistently yielded better results than the baseline CNN neural network or the OpenMax method. The improvement is 1% for the CIFAR and 6% for the MNIST datasets. In each individual test sample, the extended threshold dominated both methods mentioned above.

The tests on the Isolation Forest method has clearly shown, that from the variants of the methods tested, the isolation tree with softmax output provided

the best result. Similarly to the Extended threshold approach, the isolation tree method dominated the OpenMax engine as well. In comparison of the different feature vector variants for the isolation tree, only the softmax variant dominated the Extended threshold method.

5. Conclusion

The main goal of the presented work was to investigate the efficiency of the detection of unknown classes in the image classification problem. The situation, when the test dataset contains instances of such classes which were not present in the training set, may cause significant efficiency degradation. The methods to cope with this kind of problem are investigated under the umbrella term openset learning.

In the paper, we present the key solution approaches and also introduce two proposed method variants: Extended threshold method and the Isolation Tree method. For the tests, we used three widely popular datasets: CIFAR-10, MNIST-10, and COIL-20. The CIFAR-10 dataset contains a large amount of low-quality, small images. As can be expected, this dataset is a hard target for the open-set learning problem.

Based on the test results,we can summarize our experiences in the following points:

- The proposed Isolation Forest with softmax feature representation yielded the highest accuracy.
- The other proposed method, Extended Threshold approach secured second place in the competition.
- The baseline OpenMax outlier detection method produced results comparable to the baseline ANN neural network.
- The test results clearly demonstrate that the presence of unseen categories significantly degrades accuracy.

These experiences and test results with low accuracy values show that the detection of unknown classes is a hard problem, the known methods can provide only a slightly improvement. The further optimization of the proposed method is the next key step in our investigation.

References

[1] G. Asadollahfardi: *Water Quality Management: Assessment and Interpretation*, in: Artificial neural network, Netherlands: Springer, 2014, pp. 77–91.

[2] A. Bendale, T. E. Boult.: *Towards open set deep networks*, in: Proceedings of the IEEE conference on computer vision and pattern recognition, IEEE, 2016.

[3] M. M. Diniz, A. Rocha: *Open-set deepfake detection to fight the unknown*, in: International Conference on Acoustics, Speech and Signal Processing, IEEE, 2024.

[4] e. a. Fang Zhen: *Learning bounds for open-set learning*, in: International conference on machine learning, IEEE, 2021.

[5] C. Geng, S.-j. Huang, S. Chen: *Recent advances in open set recognition: A survey*, IEEE transactions on pattern analysis and machine intelligence 43.10 (2020), pp. 3614–3631.

[6] K. Hornik, M. Stinchcombe, H. White: *Multilayer feedforward networks are universal approximators*, Neural networks 2.5 (1989), pp. 359–366.

[7] J. M. Johnson, T. M. Khoshgoftaar: *Survey on deep learning with class imbalance*, Journal of big data 6.1 (2019), pp. 1–54.

[8] A. Krizhevsky, V. Nair, G. Hinton.: *The CIFAR-10 dataset*, https://www.cs.toronto.edu/~kriz/cifar.html, Accessed: 2025-07-15, 2009.

[9] Y. LeCun, C. Cortes, C. Burges: *MNIST handwritten digit database*, https://www.kaggle.com/datasets/hojjatk/mnist-dataset, Accessed: 2025-09-23, 2010.

[10] F. Liu, K. M. T. Tony, Z.-H. Zhou: *Isolation forestv*, in: International Conference on data mining, IEEE, 2008.

[11] F. T. Liua, K. M. Ting, Z. H. Zhou: *Isolation-Based Anomaly Detection*, ACM Trans. Knowl. Discov. Data 6.1 (2012), Article 3.

[12] A. Mahdavi, M. Carvalho: *A survey on open set recognition*, in: Fourth International Conference on Artificial Intelligence and Knowledge Engineering, IEEE, 2021.

[13] S. A. Nene, S. K. Nayar, H. Murase: *Columbia Object Image Library (COIL-20)*, https://www.kaggle.com/datasets/yupanliu999/coil-20, Accessed: 2025-09-23, 1996.

[14] M. H. Saad, A. E. Salman.: *A plant disease classification using one-shot learning technique with field images*, Multimedia Tools and Applications 83.20 (2024), pp. 58935–58960.

[15] e. a. Strani Lorenzo: *One class classification (class modelling): State of the art and perspectives*, TrAC Trends in Analytical Chemistry 183 (2020), p. 118117.

[16] e. a. Wang Wei: *A survey of zero-shot learning: Settings, methods, and applications*, ACM Transactions on Intelligent Systems and Technology 10.2 (2019), pp. 1–37.

[17] J. Xu, M. Kovatsch, S. Lucia: *Learning placeholders for open-set recognition*, in: International Conference on Industrial Informatics (INDIN), IEEE, 2021.

[18] D.-W. Zhou, H.-J. Ye, D.-C. Zhan: *Learning placeholders for open-set recognition*, in: IEEE/CVF conference on computer vision and pattern recognition, IEEE, 2021.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 140–147
DOI: 10.17048/fmfai.2025.140

Applying Tree-Based Convolutional Neural Networks to classify design patterns*

Gábor Kusper[ab], Erik Zoltán Hidi[c], Krisztián Kusper[c], Zijian Győző Yang[d], Szabolcs Márien[c]

[a]Eszterházy Károly Catholic University
kusper.gabor@uni-eszterhazy.hu

[b]University of Debrecen
kusper.gabor@inf.unideb.hu

[c]InnovITech
hidieric@gmail.com, {krisztian.kusper,szabolcs.marien}@innovitech.hu

[d]ELTE Research Center for Linguistics
yang.zijian.gyozo@nytud.elte.hu

Abstract. Automatic detection and classification of design patterns are an increasingly relevant task in modern software engineering, as it directly contributes to improving code quality, readability, and maintainability. In this paper, we propose the application of a modified Tree-Based Convolutional Neural Network (TBCNN) architecture for the recognition of GoF design patterns in Java source code. The approach leverages Abstract Syntax Trees (ASTs) as structural representations of programs, where nodes are encoded by a pre-trained embedding model that captures semantic similarities between language keywords. The resulting vectorized ASTs are processed by the TBCNN, enabling the model to learn both structural and semantic features characteristic of design patterns. For training and evaluation, we collected a dataset of Java implementations of design patterns from GitHub repositories, resulting in approximately 500–600 samples per pattern. Experimental results demonstrate high classification accuracy, with average precision, recall, and F1-scores exceeding 98% across eight design patterns. These findings confirm the viability of tree-based deep learning methods for pattern recog-

*This research was supported by the grant 2018-1.1.1-MKI-2018-00200 “Creating an automated quality assurance service with refactoring solutions”.

nition in source code. However, the model shows limitations when applied to real-world production code, likely due to the restricted representativeness of the training data, which consists mainly of educational implementations.

Keywords: design patterns, Tree-Based CNN, source code analysis

1. Introduction

Design patterns provide proven, reusable solutions to recurring design problems and are widely used to improve code quality, readability, and maintainability [3]. Despite their importance, *automatic* recognition of design patterns in source code remains challenging. Traditional approaches often rely on handcrafted rules or classical machine learning over engineered features, which are costly to maintain and tend to generalize poorly across projects and coding styles [9]. Recent advances in learning on program structure suggest that models operating directly on Abstract Syntax Trees (ASTs) can capture both syntactic and structural regularities of code [8]. Building on these insights, we revisit design pattern classification through the lens of tree-based deep learning.

Our study is motivated by the early work of Márien on the decision structures underlying design patterns [4–7], which highlights that patterns exhibit distinctive structural signatures. Tree-Based Convolutional Neural Networks (TBCNNs) have shown promise in learning from ASTs without handcrafted features [8], but their applicability to *pattern-level* classification in real-world codebases has not been systematically assessed. This leaves a gap between rule-heavy detectors and structure-aware neural models.

We address the following question:

> **RQ:** Can a modified Tree-Based Convolutional Neural Network effectively and robustly classify Gang-of-Four (GoF) design patterns in Java source code from AST representations?

We focus on eight GoF patterns that are both prevalent in practice and structurally discriminative: *Adapter*, *Bridge*, *Builder*, *Decorator*, *Null Object*, *State*, *Strategy*, and *Template Method*. We target method- and class-level manifestations as they appear in compilable Java files.

We propose a two-stage pipeline. First, we *pre-train* a keyword/token embedding model on Java code to obtain semantically informed vector representations for AST nodes. Second, we feed vectorized ASTs to a *modified* TBCNN, in which we adjust the coefficient matrix computation within the tree-based convolution to better reflect heterogeneous child-role contributions. This design aims to couple semantic token proximity with structural pattern signals.

To study feasibility at scale, we curate a pattern-labeled corpus from public GitHub repositories by querying files whose names and directory contexts indicate any of the eight target patterns. While this yields substantial coverage per pattern, many examples are educational implementations. We therefore explicitly evaluate generalization and discuss limitations stemming from dataset representativeness.

This paper makes the following contributions:

1. We formulate and evaluate a structure-aware deep learning approach for design pattern classification that operates directly on ASTs using a modified TBCNN.

2. We construct a pattern-focused Java dataset covering eight GoF patterns and report comprehensive per-pattern metrics.

3. We provide an empirical analysis highlighting strong performance on academic style implementations and outlining the generalization gap to production code, thereby charting directions for hybrid and data-centric improvements.

Our experiments indicate that the proposed model achieves high precision/recall on the curated benchmark, supporting the viability of tree-based deep learning for pattern recognition. At the same time, we observe reduced robustness on production code, underscoring the need for richer, more diverse training data and for combining learned structural features with lightweight, interpretable constraints. These observations motivate future work on dataset expansion, transfer learning, and hybrid rule+ML detectors.

Section 2 provides a concise overview of related work; Section 3 details the models, the modified tree convolution, and describes the dataset; Section 4 presents experiments and results; Section 5 concludes with limitations and future work.

2. Related work

The task of detecting and classifying design patterns has been studied from different perspectives, ranging from rule-based analysis to machine learning methods. Several surveys and empirical studies have highlighted both the importance of the problem and the challenges of building reliable detectors.

A systematic review by Yarahmadi and Hasheminejad [10] provides a comprehensive overview of design pattern detection approaches, categorizing them into rule-based, metrics-based, graph-matching, and machine learning families. Their study points out that most existing techniques rely heavily on handcrafted features or structural rules, which limit generalization to diverse coding styles and large-scale industrial systems.

Early work by Alhusain et al. [1] explored the feasibility of machine learning for design pattern recognition by using feature extraction combined with neural networks. Although pioneering, their approach was constrained by small datasets and relatively simple features, which restricted robustness. Later, Chaturvedi et al. [2] applied classical machine learning algorithms such as decision trees, support vector machines, and artificial neural networks to detect design patterns. They demonstrated that machine learning can achieve competitive accuracy, but also confirmed that feature engineering is critical and often domain-specific.

Zanoni et al. [11] investigated the integration of graph-based structural analysis with machine learning. Their MARPLE-DPD framework models design pattern instances as graphs and applies supervised learning for classification. This work represents an important step toward combining structural representations with learning techniques, yet it was mainly validated on relatively small systems.

Overall, the literature suggests that machine learning approaches to design pattern detection are promising, but existing studies are often limited by hand-crafted features, small datasets, and simplified implementations. This motivates structure-aware deep learning methods, such as Tree-Based Convolutional Neural Networks (TBCNNs), which can directly exploit Abstract Syntax Trees (ASTs) without manual feature engineering. Our work builds on these insights by applying and modifying a TBCNN architecture for pattern-level classification, aiming to better capture the structural and semantic signatures of GoF design patterns in Java source code.

3. Models

As it was mentioned earlier, we use two separate models for our problem. The architectures of these models are represented on Figure 1 and on Figure 2. The first model is used only for learning the programming language. The architecture is simple: embedding layer, fully connected layer and softmax layer.

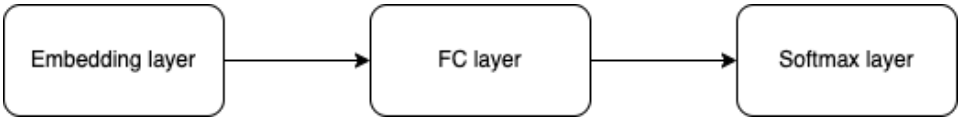


Figure 1. The keyword encoder.

The classification model has two input layers: vectorized AST and an array of node indexes. The inputs are concatenated, then the result is fed to the TBC layer, which is followed by a fully connected and a softmax layer.

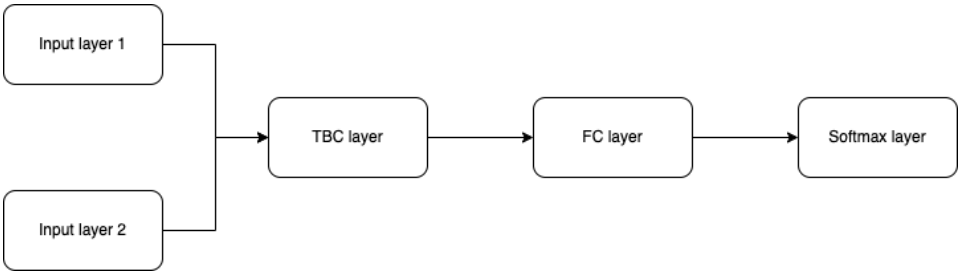


Figure 2. The design pattern classifier.

We modified the calculation of the coefficient matrix in the tree-based convolu-

tion. The equations (3.1), (3.2), and (3.3) represent the original calculations. We replaced equation (3.3) by equation (3.4).

$$\eta_i^t = \frac{d_i - 1}{d - 1} \tag{3.1}$$

$$\eta_i^r = (1 - \eta_i^t)\frac{p_i - 1}{n - 1} \tag{3.2}$$

$$\eta_i^l = (1 - \eta_i^t)(1 - \eta_i^r) \tag{3.3}$$

$$\eta_i^l = c_i \tag{3.4}$$

Where c_i is a vector with length equal to the number of children of the node filled with ones.

3.1. Training dataset

Source codes for the training dataset were gathered from GitHub, since the platform keep records of several million Java files. To select the required files, we used queries based on file names, thus downloaded Java files with names of design patterns. Approximately 500-600 files were gathered for each design pattern. The training dataset is built, but the quality may be questionable, since these files probably contain implementations for educational purposes, which makes them very similar.

4. Results

The dataset was divided into three parts: training, validation and test datasets. The proportion of files in the separate datasets is 70:15:15.

The model achieved high precision at the beginning already. After the 15th epoch, the precision is over 90% on both the training and validation datasets.

We evaluated the model successfully, the result is visible on Figures 3, 4, 5 and Table 1.

Table 1. Metrics.

DP	Precision	Recall	F1
Adapter	0.9894	1.0000	0.9947
Bridge	0.9500	1.0000	0.9744
Builder	1.0000	1.0000	1.0000
Decorator	0.9806	1.0000	0.9902
Null Object	0.9873	1.0000	0.9936
State	0.9753	0.9518	0.9634
Strategy	0.9800	0.9515	0.9655
Template Method	1.0000	0.9759	0.9878
Average	0.9828	0.9849	0.9837

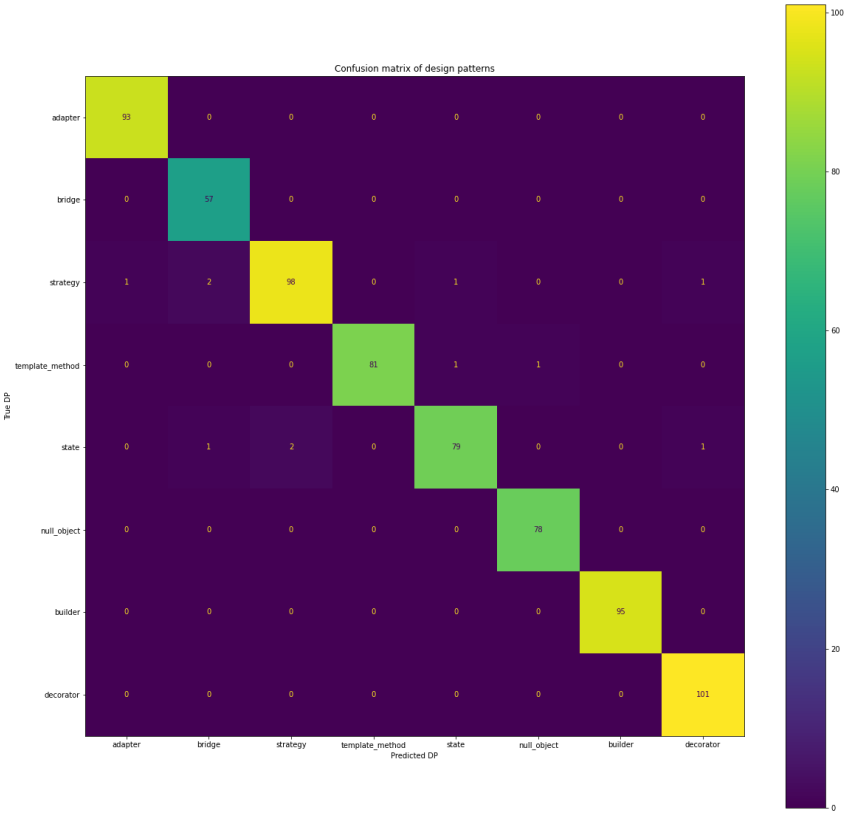


Figure 3. Confusion matrix of design pattern predictions.

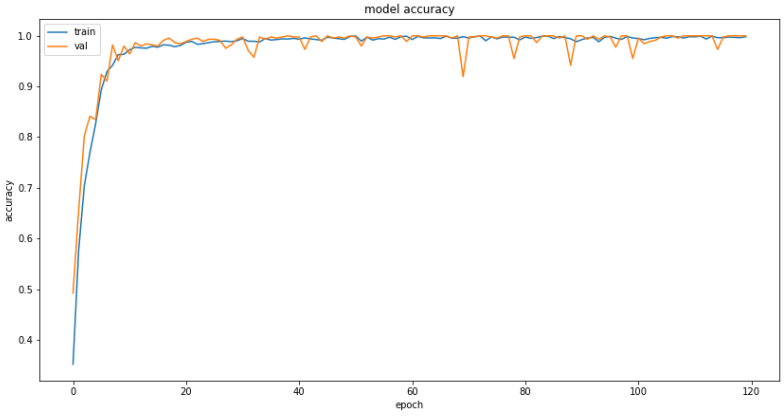


Figure 4. Training and validation accuracy across epochs.

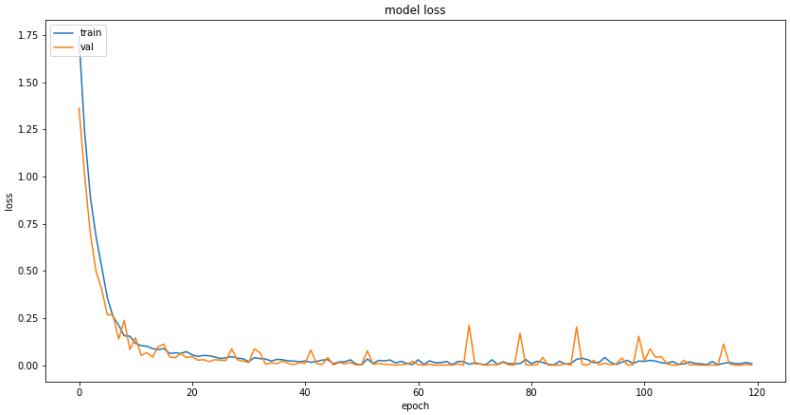


Figure 5. Training and validation loss across epochs.

5. Conclusion

In this paper, we applied a Tree-Based Convolution based machine learning model on a design pattern classification and detection problem with partial success. The model is very accurate in classifying the given design patterns, but as the experiments show it has problems with detecting design patterns in real world production code. We assume that the main problem behind this phenomenon is the lack of real world production source code in the training dataset.

Future work will focus on expanding the dataset with real-world industrial code bases and exploring hybrid approaches to improve generalization.

References

[1] S. Alhusain, S. Coupland, R. John, M. Kavanagh: *Towards machine learning based design pattern recognition*, in: 2013 13th UK Workshop on Computational Intelligence (UKCI), 2013, pp. 244–251.

[2] S. Chaturvedi, A. Chaturvedi, A. Tiwari, S. Agarwal: *Design pattern detection using machine learning techniques*, in: 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2018, pp. 1–6.

[3] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design patterns: elements of reusable object-oriented software*, USA: Addison-Wesley Longman Publishing Co., Inc., 1995, ISBN: 0201633612.

[4] S. Márien: *Decision based examination of object-oriented programming and Design Patterns*, Teaching Mathematics and Computer Science 6 (2008), pp. 83–109, DOI: 10.5485/TMCS.2008.0174.

[5] S. Márien: *Decision based examination of object-oritented methodology using JML*, in: Annales Mathematicae et Informaticae, vol. 35, Eszterházy Károly College, Institute of Mathematics and Computer Science Eger, 2008, pp. 95–121.

[6] S. Márien: *Decision structure based object-oriented design principles*, in: Annales Mathematicae et Informaticae, vol. 47, 2017, pp. 149–176.

[7] S. Márien, G. Kusper: *Understanding Design Patterns as Constructive Proofs*, Proceedings of ICAI-2004 (2004), pp. 173–182.

[8] L. Mou, G. Li, L. Zhang, T. Wang, Z. Jin: *Convolutional neural networks over tree structures for programming language processing*, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, Phoenix, Arizona: AAAI Press, 2016, pp. 1287–1293.

[9] H. Thaller, L. Linsbauer, A. Egyed: *Feature Maps: A Comprehensible Software Representation for Design Pattern Detection*, in: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019, pp. 207–217.

[10] H. Yarahmadi, S. M. H. Hasheminejad: *Design pattern detection approaches: a systematic review of the literature*, Artificial Intelligence Review 53.8 (2020), pp. 5789–5846.

[11] M. Zanoni, F. A. Fontana, F. Stella: *On applying machine learning techniques for design pattern detection*, Journal of Systems and Software 103 (2015), pp. 102–117.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 148–160
DOI: 10.17048/fmfai.2025.148

Explainable image segmentation with wavelet-network*

Hanna-Georgina Lieb, Tamás Kaszta

Babeș-Bolyai University
{hanna.lieb,tamas.kaszta}@ubbcluj.ro

Abstract. Recent advances in artificial intelligence and its widespread adoption have imposed the necessity of research targeting the inner mechanisms of intelligent systems. We lack the exact mathematical tools needed to grasp what led to a certain output. The term *explainability* has recently emerged in the context of artificial intelligence (AI) as an area of development. An efficient way to introduce a checkpoint into decision-making systems is to incorporate prototype units. These bridge the difference between input image space and feature space, offering us a glimpse into an intermediary phase of decision-making. We created a new model – WaveProtoSeg – from the WaveProtoPNet classification model, combining image segmentation with the wavelet transform as a feature extractor. Our experiments were conducted on the Cityscapes dataset, which gathers real street scenes. Although we did not achieve the accuracy of the original paper, we explored various configurations of the system, and we managed to build a versatile system.

Keywords: image segmentation, wavelet, explainability, prototype

1. Introduction

Intelligent systems are experiencing significant advances at a rapid pace and are being progressively integrated into safety-critical fields, including healthcare, defense, and autonomous driving. Despite their accuracy, deep learning models often function as black boxes, offering little insight into their decision-making processes, posing risks when human lives or assets are at stake. This has led to growing interest in explainable artificial intelligence (XAI), which aims to improve model

*This work benefitted from the project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization, and Financial Instruments Program, 2021-2027, MySMIS no. 334906.

transparency and trust [17, 19].

Prototype-based approaches have demonstrated significant potential in this field. Prototypes may be conceptualised as instances in the feature space that exemplify a certain class, encapsulating the core properties characteristic of that class, analogous to learning vector quantization [10]. They capture key features during training and serve as reference points for interpreting predictions from new data. They can reveal model errors, biases, and decision rationale, making them valuable tools for developers.

However, the implementation of prototypes poses a significant challenge, the need to connect the initial raw input with the more abstract high-level feature space. Many of the current methodologies to address this issue typically depend on complex, computationally demanding black-box architectures. In contrast, wavelet transforms [15] provide a lightweight alternative, effectively capturing spatial and frequency information from images viewed as 2D signals.

This study presents WaveProtoSeg (see Figure 2), a segmentation model that integrates wavelet-based feature extraction with interpretable prototype learning, starting from our previous work, WaveProtoPNet [12, 13]. Similarly to the ProtoSeg [18] framework, this approach focuses on achieving pixel-wise image segmentation, with the main difference in using wavelets as feature extractors. Its accuracy is assessed using the realistic Cityscapes dataset [3], which provides a rich and complex urban environment for evaluation, relevant to the needs of autonomous driving. WaveProtoSeg aims to deliver transparent and interpretable output.

The article is structured as follows. In Section 2 a review of the literature is conducted, after which the model is presented in Section 3, followed by the experiments and discussion in Section 4, ending with the conclusion in Section 5.

2. Literature review

2.1. Image segmentation and explainability

Image segmentation is a fundamental task in computer vision that assigns a class label to each pixel of an image, dividing it into semantically coherent regions. Unlike image classification, which outputs a single label per image, segmentation operates at a finer granularity, making it essential for applications like medical imaging, autonomous driving, and remote sensing [4, 22].

Although classification has historically been the entry point for XAI in research and industry, segmentation has received increasing attention since the late 2010s [25]. Early successful segmentation models include U-Net [16], designed for biomedical image analysis, and SegNet [1], which uses an encoder-decoder structure to map input images to prediction in pixels. More recently, transformer-based models such as Segmenter [21] have been introduced to capture long-range dependencies in segmentation tasks.

Segmentation poses unique challenges for interpretability. Decisions at the pixel level must account for local context and global consistency, often leading to com-

plex interactions between neighbouring pixels. This makes explanations harder to interpret and evaluate, especially when no clear ground truth is available for what constitutes a valid explanation.

Explainable image segmentation techniques can be broadly categorised into post-hoc and architecture-based methods [8]. For post-hoc methods, the model does not directly explain its predictions, and an independent model is employed to obtain this information. Architecture-based methods are inherently explainable, i.e. providing an explanation alongside the prediction. Among these inherently interpretable models, prototype-based explanations offer an intuitive and interpretable approach by associating predictions with representative examples from the training data. Each class is linked to a set of learnt prototypes, which provide visual justification for predictions based on similarity to prototypical regions [9]. Counterfactual explanations form another important branch, focusing on identifying the minimal changes to input that would alter the model output, thus helping to understand the decision boundaries and increase the robustness against adversarial perturbations. Perturbation-based methods systematically occlude or modify parts of the input image to analyse the resulting changes in output, offering insight into which regions are the most influential. Gradient-based approaches, such as saliency maps and Grad-CAM [20], utilise gradient information from subsequent network layers to produce heat maps that highlight the regions most responsible for a particular prediction [7, 24]. Finally, architecture-based methods are interpretable by design, embedding explainability directly into the model's structure rather than relying on post-hoc interpretation.

Our proposed model is similar to the previous ProtoSeg architecture introduced in [18], which demonstrated interpretable semantic segmentation through the use of prototypical image patches learnt from training data, a patch of an image being a smaller region of it. This model expands on ProtoPNet [2], which served as the basis for our earlier development of the WaveProtoPNet model. ProtoSeg incorporates a diversity loss to encourage the model to learn a broad and representative set of prototypes, thereby enhancing interpretability. In our work, we retain the core structure and interpretability framework of ProtoSeg, while extending its capabilities by integrating a wavelet-based feature extractor. This modification results in more transparent feature representations.

2.2. Wavelets in image segmentation

Wavelet transforms offer a powerful tool for analysing signals in both space and frequency domains simultaneously. Unlike the Fourier transform, which only provides frequency information and loses spatial localisation, wavelets allow multiresolution analysis, capturing both coarse structures and fine details [15].

The Discrete Wavelet Transform (DWT) decomposes a signal into approximation and detail coefficients through a series of low-pass and high-pass filters. In the context of image processing, this results in a set of sub-bands that highlight different spatial features, such as edges and textures. Wavelets are localised, meaning they are compact in both space and frequency, making them well suited for tasks

like denoising, compression, and feature extraction.

The two-dimensional wavelet transform applies these low- and high-pass filters to images by filtering along rows and columns, leading to four quarter-sized images at each decomposition level, focusing on: approximation coefficients (low+low) and detail coefficients in horizontal (low+high), vertical (high+low), and diagonal (high+high) directions. This hierarchical decomposition can be recursively applied to the approximation result for a finer scale analysis (i.e., higher decomposition levels); see Figure 1.

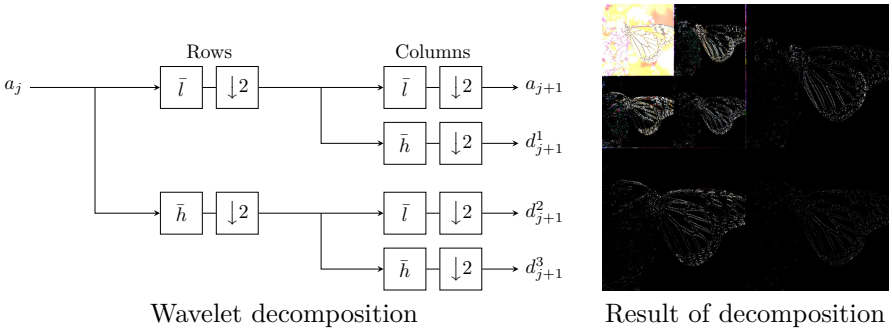


Figure 1. *(left)* Wavelet transform using low-pass and high-pass filters (l, h). *(right)* The result of the decomposition on an image. The average component (a_{j+1}, upper-left corner) underwent an additional decomposition ($\Rightarrow$ decomposition on 2 levels).

In wavelet segmentation methods, such as the Wavelet Segmentation Method (WSM) demonstrated improved performance in capturing background and small-scale features compared to classical threshold-based techniques [6]. In sonar imaging applications, wavelet filters have been used to reduce noise while enhancing feature localisation, showing the benefit of wavelet-based multiscale representations [23].

In deep learning, several models incorporate wavelet transforms into convolutional neural networks (CNNs) to replace conventional downsampling layers (e.g., max-pooling, strided convolution). For example, Haar wavelets have been used to decompose feature maps into low-frequency and high-frequency components during encoding. Integrating Haar wavelet downsampling improves segmentation accuracy, particularly in boundary regions. [26]

A notable advancement is XNet [27], a deep learning architecture that integrates DWT and Inverse Wavelet Transform into a U-Net-style encoder-decoder. XNet captures both global context and local detail by separating and recombining frequency information, leading to improved performance even under semi-supervised conditions. However, its success is limited when high-frequency features are not prominent in the input data.

In general, wavelets offer a lightweight and effective alternative for multiscale feature extraction in segmentation tasks, particularly where interpretability and

boundary precision are critical.

3. Methodology

To incorporate wavelets into the segmentation framework, we explore three distinct architectural configurations (see Figure 2), each offering different trade-offs in terms of the granularity of feature representation and output resolution.

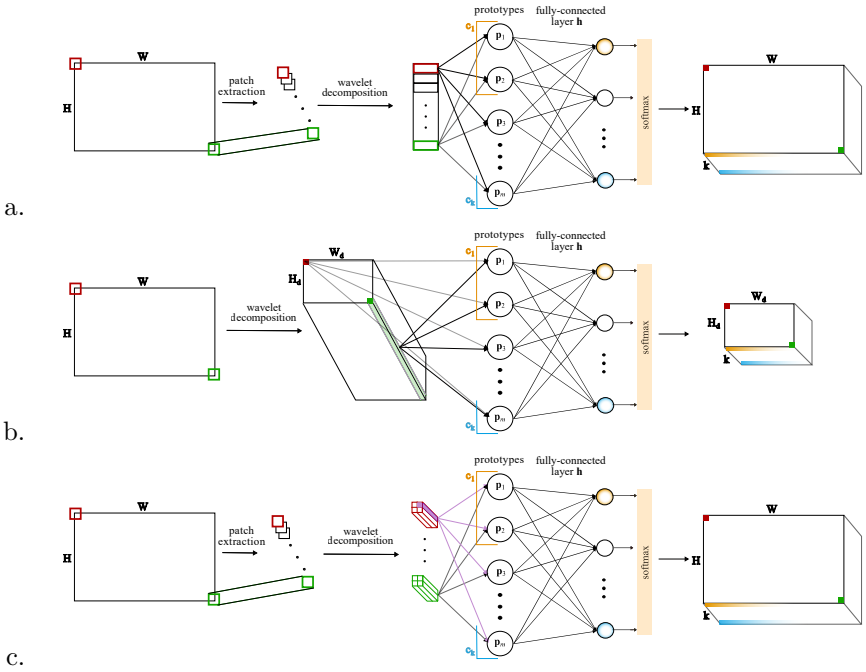


Figure 2. Different WaveProtoSeg builds: (a) patch extraction at the beginning and patch-sized prototypes, (b) patch extraction via wavelet decomposition; (c) pre-extracting patches and prototypes smaller than patch-size.

In WaveProtoSeg, image feature extraction is performed using wavelet decomposition – the different applications of it will be explained in detail in each three setups. This operation produces a set of feature maps from the input image x – denoted as $\phi(x) = z$ – which are then passed to the prototype layer. In case of *c.setup*, this will be split into smaller patches: $\tilde{z} \in \mathcal{P}(z) \equiv \text{patches}(z)$. The output produced by the j^{th} prototype unit (p_j) can be written as:

$$g_{p_j}(z) = \max_{\tilde{z} \in \mathcal{P}(z)} \log\big(\big(||\tilde{z} - p_j||_2^2 + 1\big)/\big(||\tilde{z} - p_j||_2^2 + \epsilon\big)\big).$$

Based on the similarity scores between the prototypes and the feature maps, the

classification will take place. After the classification of each pixel, the results will be gathered into one final result: the segmentation map of the original image. The prototype units themselves are also learnt from the wavelet-extracted features. The depth of wavelet decomposition, that is, the number of decomposition levels, is a tunable hyper-parameter.

In the first configuration, shown in Figure 2.a, each pixel in the input image is associated with a local patch. These patches are independently decomposed with wavelet transform until the average (low-frequency) component is reduced to a single pixel. The resulting feature map of each patch retains the same spatial dimensions as the original patch. The result is a vector of dimension $1 \times (H_p \cdot W_p)$, where $H_p \cdot W_p$ is the size of the patch. For an image x of size 32×32, this process results in 32×32 distinct patches – one for each pixel – that capture the local neighbourhood context. These feature vectors are then compared to the learnt prototypes in the prototype layer. Since this setup generates a prediction per input pixel, the output resolution matches the input. Conceptually, this is analogous to PrototypeDL [11], which learns entire images as prototypes; here, the prototype layer learns entire patches.

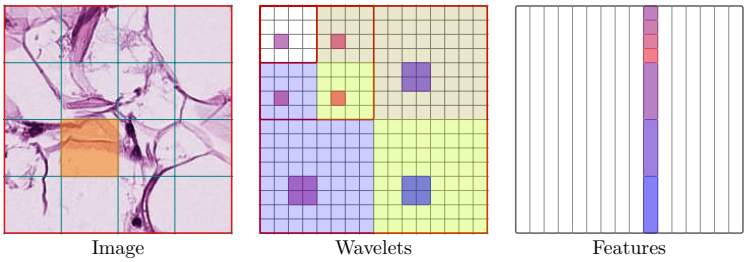


Figure 3. (left) Highlighted patch from the original image. (mid) Purple regions on the feature map indicate dispersed data of the patch. (right) Vector-form rearrangement of the feature map.

In the second configuration (*b.setup*), the entire image is wavelet-decomposed in one pass. Following decomposition, patches are extracted from the resulting feature map; see Figure 3. This significantly reduces the number of patches compared to *a.setup*, resulting in a smaller output resolution. For example, decomposing a 128×128 image may produce a 32×32 feature map, leading to 32×32 patches. Hence, the architecture in Figure 2.b provides fewer segmentation predictions, corresponding to the spatial dimensions of the decomposed map. This setup resembles the WaveProtoPNet approach, where images are decomposed into feature maps, and patches from these maps are used to learn and compare prototypes.

The third configuration (from Figure 2.c) shares structural similarities with *a.setup*, in that the patches are extracted before wavelet decomposition. However, the number and size of patches are user-defined hyper-parameters. Each patch is decomposed only up to a certain level, stopping before the approximation component reduces to a single pixel. This allows the average part to retain a spatial extent

of multiple pixels. For example, decomposing a 256×256 input into 16×16 patches and performing three levels of wavelet decomposition result in a 2×2 component for each 8×8 patch. These are flattened into $1 \times 1 \times 64$ feature vectors, which also define the prototype dimensions. This setup enables fine-grained semantic matching while maintaining computational efficiency. In this case, the prediction granularity of the model remains aligned with the input image. The inner part of this setup, after extracting patches is basically a WaveProtoPNet by structure – and its pixel-wise results will be gathered into one segmentation mask.

Despite differences in feature map construction and output resolution, all three configurations share the same prototype layer and fully connected classification layer. The size of prototype vectors varies depending on the setup, but the comparison logic remains consistent. The primary hyper-parameters include patch size, decomposition depth, and number of prototypes.

Loss functions

During training and evaluation, we adopted the loss formulation proposed in ProtoSeg [18], while also evaluating an alternative loss from our previous work, WaveProtoPNet. The training comprises two distinct phases: initially, it concentrates on prototype formation, and subsequently, it emphasizes classification accuracy while mitigating negative reasoning.

Let $D = \{(x_i, y_i)\} = [X, Y]$ denote a data set of images and labels.

In the first phase, when the focus is on prototype learning, the weights of the fully connected layer h are frozen, so that the edges that connect a prototype with the class for which they are responsible are set to 1, otherwise to -0.01:

$$w_h^{(k,j)} = \begin{cases} 1 & \text{if } p_j \in P_k \\ -0.01 & \text{otherwise,} \end{cases}$$

where $w_h^{(k,j)}$ denotes the weight of the edge connecting the j^{th} prtotoype to the class k, and P_k being the set of prototypes responsible for class k.

The loss responsible for prototype-formation from ProtoPNet includes:
Cluster Cost ($\mathcal{L}_{Clst}$), which encourages each prototype to be close to at least one patch of its corresponding class: $\mathcal{L}_{Clst} = \frac{1}{n} \sum_{i=1}^{n} \min_{p_j \in P_{y_i}} \min_{z \in \mathcal{P}(\phi(x_i))} ||z - p_j||_2^2$; Separation Cost (Sep), that promotes distance between prototypes and patches of different classes: $\mathcal{L}_{Sep} = -\frac{1}{n} \sum_{i=1}^{n} \min_{p_j \notin P_{y_i}} \min_{z \in \mathcal{P}(\phi(x_i))} ||z - p_j||_2^2$. The total loss, focusing on prototype formation, includes both the cross-entropy classification loss ($\mathcal{L}_{CE}$) and regularisation terms:

$$\mathcal{L}_1 = \mathcal{L}_{CE} + \lambda_1 \mathcal{L}_{Clst} + \lambda_2 \mathcal{L}_{Sep}$$

In contrast, the ProtoSeg loss function introduces a prototype diversity term based on Jeffrey's divergence. It will be written in a suitable form for *a.setup* and *b.setup*. This term ensures that prototypes of the same class are activated in different regions of the input image, promoting interpretability and coverage.

Jeffrey's divergence between the distributions U and V is defined as: $D_J(U,V) = \frac{1}{2}D_{KL}(U \parallel V) + \frac{1}{2}D_{KL}(V \parallel U)$, being the symmetrised version of the Kullback-Leibler divergence (D_{KL}). For multiple distributions $U_1, U_2, \ldots, U_n$, their similarity is computed as: $S_J(U_1, U_2, \ldots, U_n) = \frac{1}{C_2^n} \sum \exp(-D_J(U_i, U_j))$.

Given a prototype corresponding to class c: $p \in P_c$ and a feature map Z with the corresponding ground truth labels $Y_Z \in \mathbb{R}^{H_d \times W_d}$, the prototype-class-image activation vector is: $v(Z,p) = \text{softmax}(\parallel z_{ij} - p \parallel^2 \mid z_{ij} \in Z, Y_{ij} = c)$. The diversity loss for prototypes of class c is then: $L_J(Z, P_c) = S_J(v(Z,p_1), \ldots, v(Z,p_k))$.

The overall prototype diversity loss, averaged over all classes, is:

$$\mathcal{L}_J = \frac{1}{C} \sum_{c=1}^{C} L_J(Z, P_c),$$

Finally, the total prototype-formation loss used during training combines the cross-entropy loss with the diversity term:

$$\mathcal{L}_{1J} = \mathcal{L}_{CE} + \lambda_J \cdot \mathcal{L}_J,$$

where $\mathcal{L}_{CE}$ is the pixel-wise classification loss, and λ_J controls the weight of the diversity regularization.

The second loss in both cases is to focus on avoiding negative reasoning. They are mostly the same, with the difference that in the ProtoSeg loss, the prototype-formation loss is also included. In case of ProtoPNet loss, it looks as follows:

$$\mathcal{L}_2 = \mathcal{L}_{CE} + \lambda \sum_{k=1}^{K} \sum_{j, p_j \notin P_k} |w_h^{(k,j)}|.$$

In the ProtoSeg type, instead of the $\mathcal{L}_{CE}$ term, $\mathcal{L}_{1J}$ occurs.

4. Results and discussion

To evaluate the performance of the WaveProtoSeg model, we used a preprocessed version of the Cityscapes dataset [3], including 5000 images relevant to understanding the urban scene [14]. This data set provides high-resolution RGB street view images from various cities of size 128×256, annotated at the pixel level of the 20 classes. These fine annotations allow for a detailed analysis of semantic segmentation performance. The ProtoSeg model achieved 67% mIoU with this dataset.

We tested several variants of the model using different types of wavelets, decomposition levels, prototype numbers, and loss functions. Across all setups, we adopted the mean Intersection over Union (mIoU) metric to evaluate segmentation performance. The training included two phases. The first phase (responsible for prototype learning) has additional three learning components: first with a learning rate of 0.01 for 8 epochs, then a learning rate of 0.005 for 4 epochs, and finally with a learning rate of 0.001 for 4 epochs. The second phase (focusing on accuracy and

avoiding of negative reasoning) has also three sub-phases, with the same learning rates as in the first phase, just with different epoch numbers per each: it runs primary for 8 epochs, then for 6, then again for 8 epochs. As loss hyper-parameters the following values were used: $\lambda_1 = 1.25$, $\lambda_2 = 0.4$, $\lambda = 0.001$, $\lambda_j = 0.25$.

In the following, we present a summary of the most successful configurations of each, illustrated in Table 1. The best overall result was achieved with the *c.setup*, where dilation of convolutional filters was employed at the initial patch extraction. This led to a mIoU of 39.21% in the test set and 41.19% on the train set. Here, the loss from WaveProtoPNet was used. The second most successful setup was the *a.setup*, which is similar to the *c.setup*, with the difference that here the patches are not decomposed into smaller patches. More prototypes were needed to achieve the above 38% mIoU, using the loss of ProtoSeg, and as a consequence, training was much more time consuming. The model with the worst performance was *b.setup*, achieving the maximum test mIoU of 33.86%, applying the loss of ProtoPNet during training.

Table 1. Summary of the best results for each setup.

Setup	Receptive field of a pixel	Decomp	Wavelet	Proto/ Class	Test mIoU
a.setup	8×8	3	db2	20	38.2
b.setup	10×10	1	db4	20	33.86
c.setup	16×16	1	db4	5	**39.21**

Despite the architectural novelty, the WaveProtoSeg model underperformed compared to state-of-the-art semantic segmentation models. Figure 4 shows a visual comparison of input images, ground truth annotations, and predictions from the WaveProtoSeg model. It is evident that some classes such as *building*, *sky* or *vegetation* were learnt better, while others like *traffic light* or *human* suffered from inconsistent predictions and object-level confusion.

There are several potential causes that could contribute to the observed lower mIoU level. One potential explanation for this phenomenon is that certain classes are significantly under-represented when compared to others. The majority of images are covered with *road*, *building*, *sky* and *vegetation*, while the rest of the classes occur just occasionally, sometimes just in a really small part of the image. Another reason should be the low quality of the prototypes. As shown in Figure 5, the learnt prototypes lack semantic structure and often do not represent the meaningful features of the training data. This can be the result of the previously mentioned problem of class-imbalance. The features extracted by wavelet decomposition can also be a problem. The wavelets may not be suitable for extracting the most relevant information from this type of data. Finally, an essential drawback can be the small receptive field that is used to classify a pixel. By expanding the area considered around a single pixel, the amount of contextual semantic information increases, which can significantly enhance the precision of its classification process.

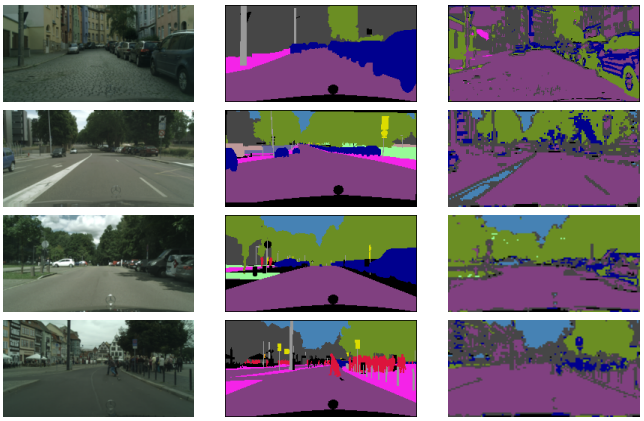


Figure 4. Top to bottom: original images, ground-truth annotations, predicted segmentation.

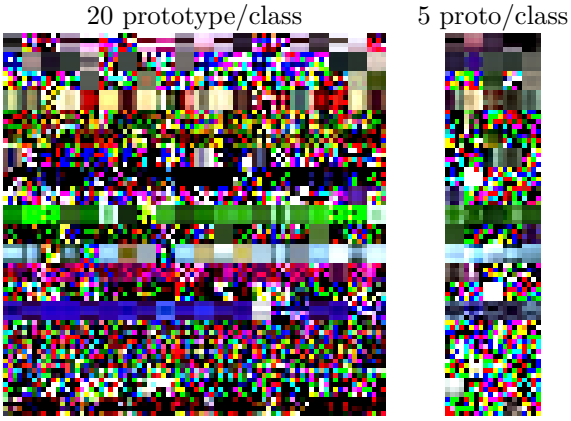


Figure 5. Visualization of learned prototypes: their low quality and poor discriminative power directly affect the model's accuracy.

5. Conclusion

In our previous research, we demonstrated that wavelets are as powerful in feature extraction as any classical backbone. Our experiments provided a thorough basis for including wavelet decomposition into prototype-based explainable systems. However, after several trials and experiments, we arrived at the conclusion that wavelet-based image segmentation remains backward compared to traditional backbone systems in terms of accuracy and interpretability.

Although the WaveProtoSeg model offers an explainable approach to semantic segmentation by combining prototype learning with wavelet-based feature ex-

traction, its performance fell significantly short of expectations on the Cityscapes dataset. The best test mIoU achieved was only 39.21%, suggesting that the current architecture and training methodology require refinement before it can be considered competitive.

The small receptive field of the model hinders its ability to make context-aware decisions. Since a pixel is classified based on features extracted from a relatively narrow patch, it lacks the broader contextual information that is often crucial in distinguishing between semantically similar regions (e.g., distinguishing a car from a bus, or a road from a sidewalk). Enlarging the receptive field may help mitigate this problem; however, this could prove to be prohibitive with respect to the number of prototypes that would be required, making the training process too expensive.

To thoroughly investigate if the model's architectural constraints are influenced by particular datasets, we intend to conduct experiments across various datasets. This includes testing on a more balanced dataset with a smaller number of distinct classes. Another direction would be to test it on a medical data set. The prior model we developed, WaveProtoPNet, demonstrated strong performance in the classification of human tissues. Inspired by these results, a worthwhile endeavour would be to investigate whether good segmentation capabilities can be achieved for medical data sets, such as the histology data set for nuclei segmentation [5]. We think that the "simpler" images – which can be converted to gray-scale without information loss – from the medical domain of the cellular level can be more convenient for prototype-based segmentation. Using these simpler data, one could experiment with the possibility of making the prototypes *rotation-invariant* by manually generating the rotated versions of the prototypes for every single learnt prototype.

In conclusion, while the model promotes explainability and novelty, substantial architectural and algorithmic improvements are necessary for it to be a viable tool for semantic segmentation in the natural or medical imaging domains.

References

[1] V. Badrinarayanan, A. Kendall, R. Cipolla: *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*, 2016, arXiv: 1511.00561 [cs.CV].

[2] C. Chen, O. Li, A. Barnett, J. Su, C. Rudin: *This looks like that: deep learning for interpretable image recognition*, in: NIPS 33, Curran Ass., 2018.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele: *The Cityscapes Dataset for Semantic Urban Scene Understanding*, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[4] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, K. Dietmayer: *Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges*, IEEE Transactions on Intelligent Transportation Systems 22.3 (2021), pp. 1341–1360, doi: 10.1109/TITS.2020.2972974.

[5] J. Gamper, N. A. Koohbanani, K. Benes, A. Khuram, N. Rajpoot: *PanNuke: an open pan-cancer histology dataset for nuclei instance segmentation and classification*, in: European Congress on Digital Pathology, Springer, 2019, pp. 11–19.

[6] J. Gao, B. Wang, Z. Wang, Y. Wang, F. Kong: *A wavelet transform-based image segmentation method*, Optik 208 (2020), p. 164123, ISSN: 0030-4026, DOI: 10.1016/j.ijleo.2019.164123.

[7] R. Gipiškis, C.-W. Tsai, O. Kurasova: *Explainable AI (XAI) in image segmentation in medicine, industry, and beyond: A survey*, ICT Express 10.6 (2024), pp. 1331–1354, ISSN: 2405-9595.

[8] R. Gipiškis, C.-W. Tsai, O. Kurasova: *Explainable AI (xAI) in Image Segmentation in Medicine, Industry, and Beyond: A Survey*, 2024, arXiv: 2405.01636 [cs.CV].

[9] S. S. Y. Kim, N. Meister, V. V. Ramaswamy, R. Fong, O. Russakovsky: *HIVE: Evaluating the Human Interpretability of Visual Explanations*, 2022, arXiv: 2112.03184 [cs.CV].

[10] T. Kohonen: *Self-Organization and Associative Memory*, Third, New York, NY: Springer-Verlag, 1989.

[11] O. Li, Liu, et al.: *Deep Learning for Case-Based Reasoning Through Prototypes: A Neural Network That Explains Its Predictions*, in: Proc. of AAAI Conf. on Artificial Intelligence, vol. 32, 2018.

[12] H.-G. Lieb, T. Kaszta, L. Csató: *On Background Classes in Prototype-Based Architectures*, Acta Universitatis Sapientiae, Informatica 17.1 (2025), p. 6.

[13] H.-G. Lieb, T. Kaszta, L. Csató: *Wavelet-Based Prototype Learning for Medical Image Classification*, in: 2024 IEEE 22nd Jubilee International Symposium on Intelligent Systems and Informatics (SISY), 2024, pp. 000631–000636, DOI: 10.1109/SISY62279.2024.10737523.

[14] S. Liu, E. Johns, A. J. Davison: *End-to-End Multi-task Learning with Attention*, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1871–1880.

[15] S. Mallat: *A Wavelet Tour of Signal Processing*, Elsevier, 2009.

[16] O. Ronneberger, P. Fischer, T. Brox: *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015, arXiv: 1505.04597 [cs.CV].

[17] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, C. Zhong: *Interpretable machine learning: Fundamental principles and 10 grand challenges*, Statistics Surveys 16 (2022), pp. 1–85, DOI: 10.1214/21-SS133.

[18] M. Sacha, D. Rymarczyk, Ł. Struski, J. Tabor, B. Zieliński: *ProtoSeg: Interpretable Semantic Segmentation with Prototypical Parts*, 2023, eprint: 2301.12276 (cs.CV).

[19] W. Samek, G. Montavon, S. Lapuschkin, C. Anders, K.-R. Muller: *Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications*, Proceedings of the IEEE 109 (Mar. 2021), pp. 247–278, DOI: 10.1109/JPROC.2021.3060483.

[20] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, D. Batra: *Grad-CAM: Why did you say that?*, 2017, arXiv: 1611.07450 [stat.ML].

[21] R. Strudel, R. Garcia, I. Laptev, C. Schmid: *Segmenter: Transformer for Semantic Segmentation*, 2021, arXiv: 2105.05633 [cs.CV].

[22] S. A. Taghanaki, K. Abhishek, J. P. Cohen, J. Cohen-Adad, G. Hamarneh: *Deep Semantic Segmentation of Natural and Medical Images: A Review*, 2024, arXiv: 1910.07655 [cs.CV].

[23] Y. Tian, L. Lan, H. Guo: *A review on the wavelet methods for sonar image segmentation*, International Journal of Advanced Robotic Systems 17 (Aug. 2020), p. 172988142093609, DOI: 10.1177/1729881420936091.

[24] K. Vinogradova, A. Dibrov, G. Myers: *Towards Interpretable Semantic Segmentation via Gradient-Weighted Class Activation Mapping (Student Abstract)*, Proceedings of the AAAI Conference on Artificial Intelligence 34.10 (Apr. 2020), pp. 13943–13944, ISSN: 2159-5399, DOI: 10.1609/aaai.v34i10.7244.

[25] K. Wickstrøm, M. Kampffmeyer, R. Jenssen: *Uncertainty and interpretability in convolutional neural networks for semantic segmentation of colorectal polyps*, Medical Image Analysis 60 (Feb. 2020), p. 101619, ISSN: 1361-8415, DOI: 10.1016/j.media.2019.101619.

[26] G. Xu, W. Liao, X. Zhang, C. Li, X. He, X. Wu: *Haar wavelet downsampling: A simple but effective downsampling module for semantic segmentation*, Pattern Recognition 143 (2023), p. 109819, ISSN: 0031-3203.

[27] Y. Zhou, H. Jiaxing, C. Wang, L. Song, G. Yang: *XNet: Wavelet-Based Low and High Frequency Fusion Networks for Fully- and Semi-Supervised Semantic Segmentation of Biomedical Images*, in: 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2023, pp. 21028–21039, DOI: 10.1109/ICCV51070.2023.01928.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 161–173
DOI: 10.17048/fmfai.2025.161

Fundamental limitations of online supervised learning in dynamic control loops*

István Pintye[ab], **József Kovács**[bc], **Róbert Lovas**[ab]

[a]Institute for Computer Science and Control, Hungarian Research Network
istvan.pintye@sztaki.hun-ren.hu

[b]Institute for Cyber-Physical Systems, John von Neumann Faculty of Informatics, Óbuda University
robert.lovas@sztaki.hun-ren.hu

[c]School of Computer Science and Engineering, University of Westminster
jozsef.kovacs@sztaki.hun-ren.hu

Abstract. In conventional supervised learning of neural networks, training samples are selected either randomly or in a predefined order, assuming independence across samples. This paper diverges from that setting by embedding the learning process within a dynamic control system. Specifically, we consider a discrete-time control system where the output is given by a nonlinear mapping, that dynamically adjusts the number of virtual machines (VMs) based on workload characteristics such as CPU, memory, and network usage. The system's output is determined by a neural network that estimates the deviation from a target utilization profile.

In online supervised learning embedded in feedback control, data generation is shaped by model performance, leading to a narrowing of the observed input distribution over time. This self-induced sampling bias may reduce model robustness, stability and adaptability. We demonstrate that simple periodic perturbations to the VM allocation process act as an effective form of regularization, improving learning robustness without relying on external reward or replay mechanisms. Unlike traditional approaches using fixed training sets, in our formulation the system operates online where at each time step, multiple candidate control inputs $u[k] \in \mathcal{U}$ are evaluated continuously and each yielding a predicted output $y[k] = f(x[k])$. At each step, the

*The research leading to these results has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101131207 (GreenDIGIT). This work is partially funded by the Hungarian Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program.

controller selects the action that minimizes the predicted deviation from the desired reference, which then determines the next state $x[k+1]$ and yields the next training sample for the neural network. As a result the learning trajectory is not predetermined but is dynamically created by the controller's actions, which depend on the network's current predictions. We present how this closed-loop interaction between prediction and sample selection influences learning stability, convergence, and input space coverage in an online setting.

Keywords: online learning, closed-loop control, neural networks, cloud resource allocation, distributional shift, adaptive systems

1. Introduction

In real-time control systems, the quality of a neural network's predictions depends not only on its internal structure and learning parameters, but also on the data it receives during training - by the trajectory of data it encounters - a trajectory shaped dynamically by its own predictions.

Control decisions are made by evaluating future system states using the current neural network model, thereby creating a prediction-driven data generation process. This feedback loop between learning and decision-making creates a unique situation where the data used for learning is not independent from the learning itself. As the model improves and control stabilizes, the diversity of training samples naturally decreases. The learning agent spends more time in well-regulated regimes, causing the network to adapt to a narrow region of the state space. This effect, which we refer to as distributional narrowing, can reduce generalization and lead the model fragile to noise, drift, or unexpected dynamics when conditions change. This phenomenon can also be viewed as a case of closed-loop covariate shift: predictions influence actions, actions influence state transitions, and the resulting data distribution becomes endogenously biased by the controller's current policy. We adopt this term to emphasize that sample selection is not external, but induced by the model-in-the-loop.

To illustrate this, we use a simulation where a system dynamically adjusts the number of active virtual machines (VMs) in response to changing workload conditions. The system receives inputs such as CPU utilization, RAM usage, and network load, and uses a neural network to predict how many VMs are needed to maintain balanced resource usage. Control decisions based on these predictions directly influence future input states, thereby closing the loop between learning and decision-making.

Our key observation is that this learning loop can be both a strength and a weakness. On one hand, the model adapts to the system behavior it helps regulate. On the other hand, it may overfit to narrow workload patterns, leading to unstable decisions under unexpected load changes. To address this, we test a simple idea: periodically apply artificial perturbations to the VM allocation decisions. These forced adjustments expose the system to underrepresented operating

regimes, increasing data diversity and improving learning robustness.

To validate these ideas, we implement a simulated resource management task in which a neural network regulates the number of virtual machines under time-varying load conditions. The task involves tracking target utilization ranges while adapting to workload dynamics. We evaluate how combinations of model complexity and artificial exploration affect learning performance. Even in a fully deterministic environment, the system's behavior varies widely depending on the diversity of training experiences encountered during learning.

In adaptive control systems, the learning process and the controller's decision-making are often interdependent. While classical supervised models assume fixed training data and offline learning, real-time systems operate under evolving conditions that require continuous adaptation. In this study, we integrate online supervised learning into a feedback-controlled system in which the learning model directly influences, and is influenced by, system behavior. Specifically, we consider a discrete-time nonlinear state-space system of the form: $y[k] = f(x[k], u[k])$, where $f(\cdot)$ is a neural network approximator trained online. Unlike classical linear systems expressed as $y[k] = Cx[k] + Du[k]$, the output here is determined by a learned nonlinear function of the current system state $x[k]$. At each time step, a finite set of candidate control actions $u \in \mathcal{U}$ is evaluated. These candidates yield different state transitions and predicted outputs. The controller selects the action that minimizes the deviation from a predefined target $y_{\text{ref}}[k]$:

$$u^*[k] = \arg\min_{u \in \mathcal{U}} \|f(x[k]; u[k]) - y_{\text{ref}}[k]\|.$$

This feedback mechanism works as the model's current performance affects which training data will be observed next, closing the loop between prediction and future training input. The system is thus self-organizing in terms of data generation and the network continuously adapts to the evolving input-output trajectory.

Through extensive simulations, we show that periodically forcing the agent into previously unseen and suboptimal regions of the state space-rather than allowing it to self-regulate exclusively based on its current policy can improve the learning process. This artificial perturbation mechanism cause greater sample diversity which leads to more robust and generalizable neural network predictions in VM allocation. We demonstrate that the same system, exposed to the same environment, may yield lower cumulative tracking errors depending on the external perturbation and exhibit more stable learning behavior compared to purely autonomous learning mechanism. As we later demonstrate, these results point to the importance of actively managing the learning process state space exploration during learning. In contrast to traditional approaches that focus on dynamic or adaptive hyperparameter optimization, we emphasize the role of sample diversity and input space coverage as one of the key determinant of control performance.

2. Related work

Much prior work in supervised learning has investigated the effect of sample selection and ordering on convergence. Curriculum learning, hard negative mining, and

stratified sampling methods are some well-known examples for these. But these typically rely on predefined sampling strategies to improve convergence speed or model robustness [1, 15]. Bouchard et al. introduce an online approach to dynamically select training examples based on the current model state, improving learning efficiency in evolving data contexts [2], which is conceptually related to our closed-loop sample selection.

General surveys such as Soviany et al. [11] provide a taxonomy of curriculum learning methods, including self-paced and RL-guided curricula, while Narvekar et al. [10] frame curriculum generation in reinforcement learning domains as a formal, adaptive process suitable for sequential decision-making settings. The curriculum learning paradigm itself has been more deeply explored in recent work. Hacohen and Weinshall (2019) systematically examine how scoring and pacing strategies affect deep network convergence, showing substantial gains in both accuracy and training speed via curriculum schedules [6]. Graves et al. (2017) propose an automated curriculum learning framework where a multi-armed bandit selects training samples to maximize learning progress, significantly accelerating learning in sequence modeling tasks [5]. Parallels our perturbation-based approach to diversify training trajectories Liu et al. actively controls the sample order to improve convergence of learning [8].

In online learning and streaming data settings, concept drift adaptation has been extensively studied. Gama et al. presented a comprehensive survey on concept drift adaptation methodologies, including ensemble and drift detection techniques [4]. Elwell and Polikar proposed incremental ensemble methods for nonstationary environments that dynamically adapt to drift [3]. Wang et al. explored mining ensemble classifiers for evolving data streams in KDD-2003 [12]. Losing et al. introduced a dual-memory architecture to manage diverse drift types in real-time systems [9].

While reinforcement learning naturally addresses closed-loop feedback dynamics and curriculum generation in RL domains has been formally framed as an adaptive process suitable for sequential decision-making [10], fewer studies [11] consider purely supervised settings where the training data distribution is influenced by the model's own predictions. Our work focuses on these feedback-coupled systems without relying on reward signals or repeated training episodes. It aligns with adaptive control and online learning under concept drift, but prioritizes one-pass supervised updates in continuous feedback systems. These works support the view that careful ordering of training examples can act as a powerful regularizer in dynamic systems.

3. Motivation

Traditional supervised learning often assumes that samples are drawn i.i.d. from a static distribution. However, in real-time systems where a model is embedded within a controller, this assumption no longer holds. Here, the learning process affects the system's state evolution, and in turn, the system determines which

samples the model sees next. In contrast, when a model is used within a control loop, the input data becomes dependent on the model's predictions. Each decision influences the system's next state, which in turn determines the next input to the model.

This feedback-induced data generation process creates both a challenge and an opportunity: model predictions directly influence control actions and resource allocation, which in turn shapes future input observations such as CPU, RAM, and network utilization. In other words, the model does not only passively receive and learns from data but also actively contributes to generating it.

This feedback structure creates a coupling between the model's estimation and the trajectory of training samples. A sequence of high-error predictions can move the system into rarely visited regions of the state space, where the model has limited prior exposure. On the other hand, consistently low-error predictions may constrain the system's behavior to a limited subset of states. In both cases, the data distribution becomes non-representative of the broader task space.

Over time, this interaction may cause the learning process to become biased toward a narrow region of the state space, making the model fragile to sensor drift, unexpected conditions, or shifts in workload dynamics. Since online learning systems typically process each sample only once and offer no opportunity to reset or rebalance the training sequence, such distributional biases cannot be corrected retrospectively.

We propose to investigate whether artificially introducing diversity during training improves performance. By occasionally overriding the model's predicted control decision with a random alternative, the system is exposed to unfamiliar states that would otherwise remain unvisited. This forced control input changes the system's trajectory and results in exposure to input regions that would otherwise be excluded. This forced exploration is expected to increase input coverage, potentially improving stability and generalization, particularly in higher-capacity models. Our goal is to evaluate how such a modification affects the convergence, generalization, and stability of the learning process under different model capacities.

The central motivation of this work is to understand how this coupling influences: (1) the speed and stability of convergence, (2) the diversity of encountered workload regimes, and (3) the robustness of learning under dynamically shifting data distributions. Unlike curriculum learning or sample-weighting strategies that assume external control over data order, our method embeds sampling decisions within the system dynamics and controller behavior. This fully endogenous sample selection process also presented by [14] illustrating the value of dynamic sampling policies under evolving data distributions.

4. System model

To explore the learning dynamics discussed above, we employ a simplified simulation model which captures the essential features of real-time online learning in control systems. We consider a discrete-time control system (in which) where

an agent (e.g., a simplified autoscaler) dynamically adjusts the number of virtual machines (VMs) in response to synthetic workload signals generated along a time-varying pattern with increasing frequency and amplitude. The system dynamics are modeled in two parts: a linear state transition model and a nonlinear output approximation via a neural network. This workload stresses the spectral resolution of the estimator. The smaller MLP (5, 3) tends to underfit higher-frequency components, while the larger MLP (10, 5) has sufficient capacity to fit them but, in autonomous mode, may over-specialize to frequently visited regimes. Periodic perturbation exposes rarer, high-frequency regions, improving coverage and stabilizing learning, consistent with our capacity proxy 101^2 vs. 42^2.

Let the state vector at time k be denoted as $x[k] \in \mathbb{R}^n$, which consists of the current sensor readings (e.g., CPU usage, RAM usage, Network usage). The control input $u[k] \in \{-10, -9, \ldots, 10\}$ represents the agent's discrete VM number adjustment at each time step. The state transition is modeled by the linear system:

$$x[k+1] = Ax[k] + Bu[k], \tag{4.1}$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^n$ are system matrices that approximate how the state evolves based on previous measurements and applied input.

The output variable $y[k] \in \mathbb{R}$ represents the deviation of the current system load state from the desired utilization target (e.g., balanced CPU and network usage), it is estimated through a neural network mapping the predicted state to a scalar value:

$$y[k] = f(x[k]; \theta), \tag{4.2}$$

where $f(\cdot; \theta)$ is a fully connected feedforward neural network with parameters θ, trained online using incoming sensor data and revealed ground truth offsets. The primary goal of the control system is to maintain resource usage close to a predefined reference profile, minimizing deviations from the target CPU and network utilization levels. Both the neural network parameters θ as well as the linear transition matrices A and B are adapted continuously during execution. The matrices A and B are estimated online using least-squares regression over observed pairs of consecutive system states and applied control inputs.

At each time step, all possible control inputs $u \in \mathcal{U} = \{-10, \ldots, 10\}$ are evaluated by simulating the next state and passed through the neural network to predict the resulting deviation. At each step, all possible actions are simulated using the previously defined dynamics in equations (4.1) and (4.2), to evaluate the predicted next state and output deviation. The control input $u^*[k]$ is selected to minimize the distance from the reference:

$$u^*[k] = \arg\min_{u \in \mathcal{U}} \|\tilde{y}[k; u] - y_{\text{ref}}[k]\|. \tag{4.3}$$

This makes the data-generating process dependent on the model's own predictions, creating a self-reinforcing learning trajectory. After each control decision, the system also transitions to a new state $x[k+1]$, determined by the internal dynamics (which is approximated), and a new data pair $(x[k], y[k])$ is added to the

training set. After executing $u^*[k]$, the true output error is revealed as the actual deviation of the system from the target utilization, computed based on workload indicators such as CPU or network saturation, denoted $y_{\text{true}}[k]$. This value is used to update the neural network via online stochastic gradient descent:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(f(x[k]; \theta), y_{\text{true}}[k]),$$

where θ are the neural network parameters, η is the learning rate set to 0.01, and $\mathcal{L}$ is the loss function (in this case mean squared error).

As we can see this procedure sequence creates a unique learning loop. Model predictions influence control actions, which influence state transitions, which then determine future training data. The interplay of learning and control thus forms an implicit curriculum, driven not by external design but by internal performance.

In this setup, the learning rate η is not tuned during operation and is fixed throughout the entire run. Its value strongly affects both the convergence speed and stability of the prediction model and indirectly, the control behavior of the agent. Since no replay buffer or offline tuning is available, the system's learning trajectory is entirely shaped by the chosen hyperparameters at initialization. But this study does not aim to optimize learning rate dynamics. Once a suitable learning rate was empirically selected, it remained fixed throughout all training runs, allowing us to isolate the effect of perturbations and model capacity on learning behavior.

We investigate: 1. How this learning process converges toward an optimal solution over time. 2. Whether it enables more efficient exploration of the state space. 3. The trade-offs between control accuracy and learning robustness under online adaptation. 4. Whether the learning process becomes more stable when the agent is periodically forced into previously unseen states. For example by artificially perturbing the VM scaling decisions at regular intervals to improve generalization

5. Methods

5.1. Simulation environment

To investigate the online learning behavior of a closed-loop neural controller, we developed a simulation environment in Python using the Pytorch framework. The environment models a simplified cloud infrastructure where a controller dynamically adjusts the number of active virtual machines (VMs) in response to changing resource demands. The simulated system operates under a synthetic workload pattern that varies over time, mimicking real-world fluctuations in service usage. At each discrete time step k, the agent observes current resource utilization and selects a control input $u[k] \in \{-10, ..., 10\}$, which corresponds to scaling the VM pool up or down by discrete steps. At every time step, the agent receives three resource utilization metrics: CPU utilization percentage, RAM (memory) utilization percentage and Network bandwidth usage. These readings are normalized and form the input state vector:

$$x[k] = \begin{bmatrix} \text{CPU}[k] & \text{RAM}[k] & \text{NET}[k] \end{bmatrix}^\top,$$

which serves as input both for the control dynamics (see Equation (4.1)) and the neural network output estimation (see Equation (4.2)).

5.2. Training procedure

At each time step, the control loop proceeds as follows:

1. The current state vector $x[k]$ is recorded from system metrics.
2. For each candidate scaling action $u \in \mathcal{U}$:
   (a) Simulate the next system state $\tilde{x}[k+1;u] = Ax[k] + Bu$, reflecting how scaling the VM pool affects resource usage.
   (b) Estimate the deviation from the target operational balance via $y = f(\tilde{x}[k+1;u])$.
3. Choose $u^*[k]$ that minimizes the deviation from the desired output $y_{\text{ref}} = 0$.
4. Apply $u^*[k]$, observe the resulting true deviation $y_{\text{true}}[k]$, and transition to the next state $x[k+1]$.
5. Update the neural network via online stochastic gradient descent. The linear dynamics matrices A and B are also updated using least-squares regression to approximate the effect of control actions on future state transitions.

The neural network function $f(\cdot)$ is modeled as a fully connected feedforward neural network with hyperbolic tangent activations. It maps the current three-dimensional system state $x[k]$ to a scalar estimate of the deviation from balanced resource usage y. For a two-hidden-layer MLP with hidden sizes (h_1, h_2), input dimension $d = 3$, and scalar output $m = 1$, the number of trainable parameters is $W = dh_1 + h_1 + h_1h_2 + h_2 + h_2m + m$. In our two configurations this gives: (10, 5) $\rightarrow W = 101$; (5, 3) $\rightarrow W = 42$. For smooth activations such as tanh, classical VC/pseudodimension bounds scale polynomially in W; in practice we report W^2 as a coarse capacity proxy: $101^2 = 10{,}201$ vs. $42^2 = 1{,}764$. We use these values only to compare relative capacity between settings and capacity alone does not guarantee online generalization in closed-loop learning. The loss is defined as the squared error between predicted and true deviation: $\mathcal{L}(f(x[k]), y_{\text{true}}[k]) = (f(x[k]) - y_{\text{true}}[k])^2$.

Learning occurs online – one sample at a time – without replay buffers or mini-batching. The system operates under a one-pass constraint, reflecting real-world adaptive control environments where replay is infeasible.

6. Experiments

As defined in Equation (4.3) in our simulated environment, the system adjusts the number of active virtual machines (VMs) based on current workload indicators: CPU utilization, RAM usage, and network traffic. At each time step, several candidate VM counts $u[k] \in \mathcal{U}$ are evaluated using the neural network model $f(x[k], u[k])$, which predicts the deviation from a target utilization profile. The system selects the VM count that minimizes this predicted deviation (see Equation (4.3)).

To investigate the role of extra perturbation, we conduct multiple simulation runs with different random seeds across identical environments. The primary control objective is to maintain a balanced utilization profile across CPU and network resources. The true deviation from this balance is denoted by $y_{\text{true}}[k]$, which reflects the difference between observed and target utilization ratios. To evaluate learning performance, we conducted multiple simulation runs under each experimental condition and measured cumulative tracking error, the sum of absolute deviations from the target resource utilization profile over time. At each timestep k, the agent's distance from the optimal trajectory is measured and added to the total error. We report both the mean and standard deviation of this metric across runs. Formally, this is defined as $E^{(i)}_{\text{track}} = sum^{T}_{k=1}\left|y^{(i)}_{\text{true}}[k] - y_{\text{ref}}[k]\right|$, and the summary statistics are computed as

$$\mu_{\text{track}} = \frac{1}{N}\sum_{i=1}^{N} E^{(i)}_{\text{track}} \quad \text{and} \quad \sigma_{\text{track}} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(E^{(i)}_{\text{track}} - \mu_{\text{track}}\right)^2}.$$

These metrics provide insight into both the control accuracy (via tracking error) and the learning performance of the model (via prediction error), enabling a robust comparison between the perturbed and non-perturbed training regimes.

7. Results

To further investigate the interplay between model complexity and input space diversity in online neural control, we conducted a set of four experiments combining two architectural settings and two training modes. The neural network used to estimate the system's deviation from target resource balance was configured with either a smaller MLP (5, 3) or a larger MLP (10, 5) structure. Each configuration was trained under one of two regimes. (A.) Autonomous control, where the agent always selected the action predicted to be optimal. (B.) Artificially disturbed control, where every third decision step (from time 0 to 1500) was randomly overridden with a uniformly sampled control value from [-10, +10], while all other steps followed the prediction-based policy. We study pre/during/post phases to assess degradation after perturbation removal. Beyond "every 3 steps (0–1500)", we evaluate schedules with periods $\{2, 5\}$ and truncated windows $[0, T_0]$ with $T_0 \in \{800, 1200\}$. Denser schedules increase coverage but may introduce short-term bias; sparser schedules yield smaller bias but less variance reduction. We report median/mean cumulative error and failure rate across seeds to locate a practical trade-off.

This 2×2 setup allowed us to test the following hypotheses: (1.) The smaller network (5,3) will underperform in tracking accuracy regardless of disturbance, due to limited expressive power. (2.) The larger model (10,5) will achieve better accuracy but may exhibit instability and divergence during training. (3.) Artificial perturbation will not help the smaller network due to its limited capacity, and might degrade convergence. (4.) Larger models would benefit from exploration,

improving stability and generalization. Each experimental condition was evaluated over 100 independent simulation runs, with only the random initialization of network weights varied. The results summarize the cumulative tracking error during the test sequence. For better understanding first we present the comparision of Configuration 3 and Configuration 4 results. Instead of visualizing each trajectory individually of each run, we show the mean resource adjustment trajectory along with one standard deviation over time, providing an aggregated view of tracking performance and learning stability.

Figure 1 shows the agent's deviation from the desired operational target across multiple independent simulation runs. The results demonstrate that artificial exploration reduces variance and improves stability, especially for larger models. In contrast, autonomous learning without perturbation leads to greater performance divergence. Smaller networks show capacity limitations in all scenarios.

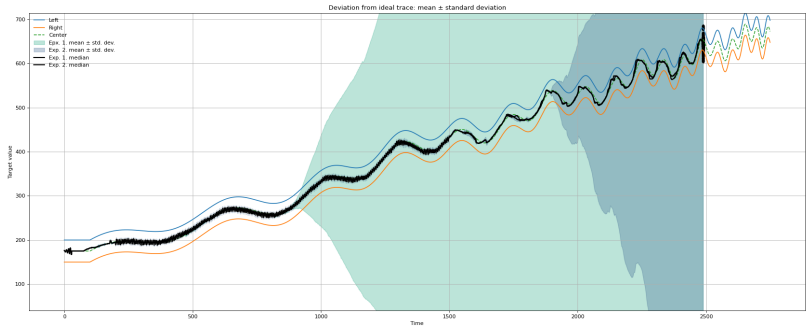


Figure 1. Error Deviation over Time for Configurations 3 and 4.

Figure 2 shows the cumulative tracking error over time on a log-scale for all four configurations. The curves are plotted on a log scale due to the large variance between successful and failed runs. In each subplot, individual runs are represented as light blue lines, with bold lines indicating the median and average trajectories. The left subplots show the cases where the agent learned and operated fully autonomously, always selecting the action deemed optimal by the model at that moment. The right subplots show the outcomes when the agent was artificially perturbed every 3 steps by forcing it into a less familiar region of the state space.

We can interpret our results as follows: Without perturbation in Configuration 1. (5,3 no perturbation) high error variance, frequent failure cases, unstable convergence occured. The network quickly overfits to a narrow input regime and fails to recover from errors. The same has happened in Configuration 3. when perturbation was not applied. Configuration 2. (5,3 + perturbation): Slight improvement in median performance, but lower failure rate even during the training. Configuration 4. (10,5 + perturbation): The larger network benefited the most from the perturbation showed better performance right after the training phase, but the performace graudally degrade again if perturbation is not applied furthermore as it can see on the bottom right plot after the 400th timestep. Cumulative

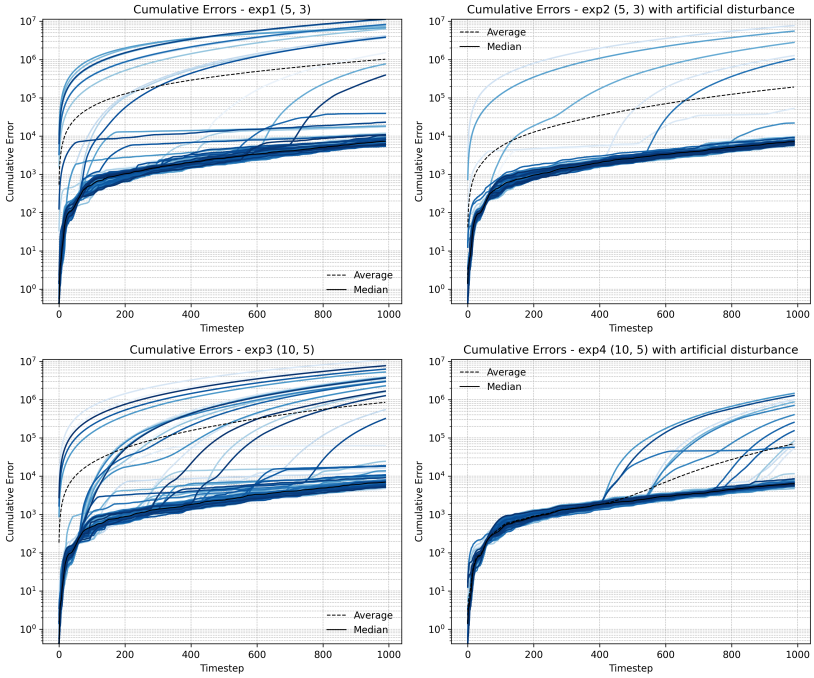


Figure 2. Cumulative error over test sequence under two type of training. Left: autonomous control with no intervention. Right: periodic perturbation to increase state-space diversity.

tracking error was recorded over time for all runs. Additionally, we defined a simple failure condition: a run was considered failed if, by the final timestep, the agent's deviation from the operational target exceeded a critical threshold. A run was classified as a failure if its final cumulative tracking error exceeded 20 000. This threshold was chosen empirically by observing that successful runs typically remained below 10 000 at the final timestep, while unstable runs accumulated errors an order of magnitude larger. Thus, the threshold reliably separates stable from divergent trajectories. This indicated that the agent entered an unstable resource state from which it could not recover. This indicated that the agent entered an unstable resource state from which it could not recover. The following failure rates were observed: Configuration 1 (5, 3, no perturbation): 21 / 100, Configuration 2 (5, 3, with perturbation): 8 / 100, Configuration 3 (10, 5, no perturbation): 26 / 100, Configuration 4 (10, 5, with perturbation): 13 / 100. The comparison shows that the perturbation strategy led to more consistent behavior and reduced the variance in deviations across different runs. These results indicate that the perturbed learning process achieved more robust generalization and improved stability over time.

8. Discussion and conclusion

This paper examines the effect of this self-induced sampling bias on learning robustness and looks into a simple but useful countermeasure, which is periodic artificial perturbation of the control decisions. By occasionally forcing the system into suboptimal or rarely visited states, we aim to reintroduce input diversity and stabilize the learning trajectory. Our hypothesis just as suggested by [7] is that such artificial exploration acts as an implicit regularizer, particularly beneficial in high-capacity models prone to overfitting and instability. Intuitively, the perturbations enlarge the effective sample size in the relevant regions of the state space, counteracting endogenous sampling bias and delaying premature consolidation to a narrow operating regime. Based on our experiments, models trained with this perturbation approach reach more stable convergence and show lower long-term variance in prediction and control error. This supports the idea that periodic and focused exploration, even in supervised control systems, can work as a regularizer that helps with generalization and stability.

In this study, we have identified a structural limitation of online supervised learning when embedded in closed-loop control systems. As the model gets better at keeping the system near optimal state, the diversity of its training data inherently shrinks. Over time, this feedback loop causes the input data to become more uniform, leading to self-induced overfitting to a narrow and stable operating regime. This distributional narrowing reduces the model's ability to generalize, and makes it more vulnerable to unexpected changes or unmodeled noise, a challenge also observed in semi-supervised one-pass learning under distribution shift [13].

We showed through extensive simulations that periodic artificial perturbation, without reward, curriculum design, or replay, improves the stability of the learning trajectory. This intervention can act as a form of implicit regularization, maintaining diversity in the input space and reducing run-to-run variance in control performance, including improved stability in VM allocation tasks. These findings suggest that in feedback-coupled supervised systems, where the learner influences its own data stream, exploration could be valuable and advantageous.

As a next step, we consider that such a control mechanism would be worth studying and applying not only in simulation but also in real-world cloud infrastructure, where incoming workload and task profiles change continuously, requiring adaptive strategies that preserve robustness and optimize energy efficiency under nonstationary conditions.

References

[1] G. Alain, Y. Bengio: *Understanding intermediate layers using linear classifier probes*, arXiv preprint arXiv:1610.01644 (2016).

[2] G. Bouchard, T. Trouillon, J. Perez, A. Gaidon: *Online Learning to Sample*, in: Advances in Neural Information Processing Systems, 2015.

[3] R. Elwell, R. Polikar: *Incremental learning of concept drift in nonstationary environments*, in: IEEE Transactions on Neural Networks, vol. 22, 10, 2011, pp. 1517–1531.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia: *A survey on concept drift adaptation*, ACM Computing Surveys (CSUR) 46.4 (2014), pp. 1–37.

[5] A. Graves, M. G. Bellemare, J. Menick, R. Munos, K. Kavukcuoglu: *Automated Curriculum Learning for Neural Networks*, in: Proceedings of the 34th International Conference on Machine Learning (ICML), 2017.

[6] G. Hacohen, D. Weinshall: *On the Power of Curriculum Learning in Training Deep Networks*, arXiv preprint arXiv:1904.03626 (2019), doi: 10.48550/arXiv.1904.03626.

[7] *Handling Out-of-Distribution Data: A Survey*, IEEE Transactions on Knowledge and Data Engineering (2025), doi: 10.1109/tkde.2025.3592614.

[8] Y. Liu, C. Zhou, P. Zhang, Z. Li, S. Zhang, X. Lin, X. Wu: *CL4CO: A Curriculum Training Framework for Graph-Based Neural Combinatorial Optimization*, in: 2024, pp. 779–784, doi: 10.1109/ICDM59182.2024.00092.

[9] V. Losing, B. Hammer, H. Wersing: *Self-Adjusting Memory: How to Deal with Diverse Drift Types*, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), 2017, pp. 4899–4903, doi: 10.24963/ijcai.2017/690.

[10] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, P. Stone: *Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey*, ArXiv abs/2003.04960 (2020).

[11] P. Soviany, R. T. Ionescu, P. Rota, N. Sebe: *Curriculum Learning: A Survey*, International Journal of Computer Vision 130 (2021), doi: 10.1007/s11263-022-01611-x.

[12] H. Wang, W. Fan, P. S. Yu, J. Han: *Mining concept-drifting data streams using ensemble classifiers*, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 226–235.

[13] H. Xie, X. Liu, L. Guo: *Semi-supervised One-pass Learning under Distribution Shift*, Proceedings of the 2023 6th International Conference on Big Data Technologies (2023), doi: 10.1145/3627377.3627446.

[14] B. Zhao, L. Wang, Z. Liu, Z. Zhang, J. Zhou, C. Chen, M. Kolar: *Adaptive Client Sampling in Federated Learning via Online Learning with Bandit Feedback*, Journal of Machine Learning Research 26.8 (2025), pp. 1–67.

[15] P. Zhao, T. Zhang: *Accelerating Minibatch Stochastic Gradient Descent using Stratified Sampling*, arXiv preprint arXiv:1405.3080 (2014), arXiv:1405.3080, doi: 10.48550/arXiv.1405.3080.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 174–187
DOI: 10.17048/fmfai.2025.174

Physics-informed neural networks for acoustic wave propagation*

Marek Ružička, Martin Štancel, Miroslav Imrich, Dmytro Havrysh

Technical university of Košice,
Department of Computers and Informatics
{marek.ruzicka,martin.stancel,miroslav.imrich}@tuke.sk
dmytro.havrysh@student.tuke.sk

Abstract. Acoustic wave propagation plays a fundamental role in various scientific and engineering disciplines, including medical imaging, seismology, and acoustics. Traditional numerical methods such as the Finite Element Method (FEM) and Finite Difference Method (FDM) are widely used to model these waves [5, 24], but they often suffer from computational inefficiencies, especially for high-dimensional problems or complex geometries. This work explores the application of Physics-Informed Neural Networks (PINNs) as an alternative approach, leveraging deep learning to solve wave equations efficiently [14]. PINNs integrate physical laws directly into the neural network's loss function, enabling solutions that adhere to the governing differential equations. We present a comparative analysis of PINNs with traditional numerical solvers, highlighting advantages, limitations, and potential improvements. Our experiments demonstrate that PINNs can effectively model wave propagation with comparable accuracy while reducing computational cost in certain scenarios.

1. Introduction and related work

Acoustic waves play a crucial role in various fields, including engineering, medicine, geology, and many others [2]. Predicting their propagation is an important task that aids in solving diverse practical problems. However, this task is challenging

*The research was partially supported by the Slovak Academy of Sciences under Grant VEGA 1/0685/23, and partly by the Slovak Research and Development Agency under the project APVV SK-CZ-RD-21-0028.

due to the large number of factors influencing acoustic wave propagation.

PINNs [14] represent a recent direction in artificial intelligence that combines physics principles with machine learning to tackle complex problems. While traditional neural networks excel at uncovering complex data relationships, they often require extensive empirical data and may disregard the inherent physical constraints of the systems being studied. This can lead to inaccuracies or even incorrect predictions, especially where physical adherence is critical. PINNs address this by integrating physical laws as constraints directly into the network architecture and loss function, allowing the model to learn from data while respecting fundamental physics. They are particularly effective for problems described by partial differential equations (PDEs) or ordinary differential equations (ODEs).

Key benefits of PINNs include the integration of physical laws, such as conservation of mass, energy, or momentum, ensuring the correctness and reliability of predictions in physical tasks. They possess the ability to model physical processes with limited data by leveraging built-in physical principles, compensating for information scarcity. Furthermore, PINNs can be used for solving both forward and inverse problems, where unknown system parameters can be determined based on observed data. PINNs merge the statistical power of neural networks with the credibility of physical principles, providing results that are accurate and physically consistent, which is exceptionally important for scientific and engineering tasks, including the modeling of acoustic processes [11].

Acoustic wave propagation is characterized by parameters such as speed, frequency, and amplitude [28]. Key phenomena include non-linear distortion, which generates harmonics (e.g., in ultrasound), and dispersion, where wave speed varies with frequency. Dispersion is minor in gases and liquids but significant in solids due to frequency and directional dependencies. Wave interactions with obstacles involve diffraction and scattering, which are critical for modeling wave behavior in complex environments.

The use of PINNs for acoustic wave propagation prediction is still an insufficiently explored area. This work aims to investigate the possibilities of utilizing PINNs for this task. The objective is to first understand the domain of acoustic waves and analyze the current state of PINNs. Subsequently, a suitable neural network architecture and geometry will be implemented using the NVIDIA Modulus Sym environment for one-, two-, and three-dimensional problems. Experimental comparisons of various neural networks and differential operators will be conducted to identify the most effective configurations. Finally, the results will be evaluated and compared with classical numerical methods and analytical solutions. Research utilizing methods like FEM for acoustic processes, such as solving the Helmholtz equation using PINNs, has shown effectiveness in reproducing complex wave behavior in heterogeneous media and enclosed spaces with obstacles, even with minimal data and complex geometries. It has been already shown that the PINNs can improve simulation efficiency and accuracy, reducing reliance on extensive mesh generation and computational resources for wave propagation [15]. Our research specifically analyzes acoustic waves.

Recent work has begun to apply PINNs specifically to acoustic wave propagation and related audio or vibration-field problems. Yokota *et al.* [27] developed a PINN framework for acoustic resonance analysis in one-dimensional acoustic tubes, handling both forward and inverse cases to estimate energy loss terms. Schoder and Kraxberger [16] demonstrated the feasibility of solving the 3D Helmholtz equation using PINNs and benchmarked against FEM and analytic solutions, showing promise for forward acoustic modeling in spatially complex domains. Wang *et al.* [21] addressed scattered acoustic fields around complex structures via PINNs, which relates closely to modeling diffraction and obstacle interactions. Other applications include acoustic field reconstruction from limited or noisy measurements, such as in ducts or tube geometries [10], and ultrasound-based inverse problems to detect defects in media using PINNs informed by the acoustic wave equation [18]. These works illustrate both the potential and current limitations of PINNs: handling high-frequency components, enforcing absorbing or reflecting boundary conditions, computational cost in 3D, and robustness under noisy or partial observation.

2. Definition of PINN

PINNs are deep learning models that integrate known physical laws, typically expressed as PDEs or ODEs, into the training process to ensure that the model's predictions adhere to the underlying physics [14]. PINNs consist of three primary components: a neural network that approximates the solution of a physical system, a set of differential equations representing the physical laws governing the system, and a loss function that penalizes deviations from both observed data and physical consistency. The neural network, commonly implemented as a multilayer perceptron (MLP), approximates a target function $u(x)$ with $u_\theta(x)$, where θ denotes the model parameters such as weights and biases [4]. These networks typically employ activation functions like *tanh* or *ReLU*, with a preference for differentiable functions to facilitate effective gradient-based optimization.

To embed physical principles into the model, PINNs include the governing equations directly into the loss function. Instead of relying on numerical solvers, the network learns to satisfy physical constraints by minimizing the error induced by these equations [14]. This approach enables PINNs to generalize from limited data while maintaining physical plausibility.

Physical constraints can be introduced in two main ways. First, physical parameters, such as wave speed, pressure, or medium density, can be included as additional features in the input. This helps the network better model spatial or temporal dependencies, particularly in dynamic systems like acoustic waves [7]. Second, the physical equations themselves can be encoded into the loss function, ensuring that the learned solution remains consistent with the physics throughout the domain. The total loss function $L(\theta)$ used to train a PINN is composed of two parts: a data loss term $L_{\text{data}}(\theta)$ that measures the discrepancy between model predictions and available measurements, and a physics loss term $L_{\text{physics}}(\theta)$ that

penalizes violations of the governing physical laws [14]:

$$L(\theta) = L_{\text{data}}(\theta) + L_{\text{physics}}(\theta)$$

3. Theorem

Let $N(u)$ represent a differential operator derived from the physical law such that the governing equation is $N(u) = 0$. The physics loss term can then be defined as the mean squared error of the residual evaluated at M collocation points $\{x_j\}_{j=1}^{M}$ [14]:

$$L_{\text{physics}}(\theta) = \frac{1}{M}\sum_{j=1}^{M}|N(u_\theta(x_j))|^2$$

This term ensures that the network's outputs not only fit the data but also satisfy the physical constraints within the problem domain [3, 6]. The relative weight between the data and physics losses can be adjusted adaptively to improve training stability and accuracy.

Training involves minimizing the total loss function using optimization algorithms such as gradient descent, Adam, or L-BFGS. Additionally, techniques such as adaptive sampling, weighting strategies, and residual-based refinement can improve convergence, especially when solving stiff or ill-posed problems [11].

Beyond forward modeling, PINNs can also solve inverse problems, such as identifying unknown parameters or reconstructing missing input data from limited measurements [14]. Regularization strategies, including activation function tuning and weighted residuals, can enhance accuracy. The spatial distribution of training points, whether uniform, random, or adaptive, also influences model performance [23].

To evaluate a PINN's accuracy and physical validity, standard metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the L_2 norm are commonly used [14, 22]. Adherence to conservation laws, like mass or energy conservation, further confirms the reliability of the model [6, 20, 23].

While traditional PINNs rely on fully connected MLPs, alternative architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been employed to better handle spatial and temporal dependencies [3, 4, 17]. Recently, Bayesian extensions of PINNs (B-PINNs) have emerged as a promising approach for uncertainty quantification in scenarios where prediction reliability is critical [25].

4. Application of PINNs to wave problems

The wave equation is a fundamental equation in mathematical physics, describing a wide range of wave processes from 1D string vibrations to 2D membrane oscillations

and 3D acoustic waves. The core idea remains the same regardless of dimension: examining the dynamics of a wave propagating through a medium, reflecting from boundaries, or interacting with them. PINNs were applied to wave problems of different dimensions within the NVIDIA Modulus Sym environment.

The NVIDIA Modulus Sym framework is a specialized tool designed for modeling physical processes using PINNs. It supports GPU optimization for high computational efficiency, crucial for large-scale simulations [12]. Modulus offers built-in support for physical equations like the Helmholtz equation for acoustics and provides tools for creating and training PINNs by including physical constraints in the loss functions. While general frameworks like TensorFlow and PyTorch can implement PINNs, Modulus provides specialized tools for handling physical equations, which might require manual configuration in the others [1, 13]. Modulus allows control over simulations via configuration files, specifying training parameters, network architecture, equations, domains, and constraints.

Key advantages of NVIDIA Modulus include its optimization for NVIDIA GPUs, enabling significantly faster model training for large-scale physical simulations [12]. It features integrated tools for working with physical laws and equations, simplifying the creation of complex models with physical constraints, including the Helmholtz equation for acoustics. Modulus also provides ready-made templates and examples for various physical tasks. Disadvantages include its dependence on NVIDIA GPU hardware and the requirement for prior experience with GPUs and physical process modeling for setup and use.

We have applied PINNs to three wave problems. 1D Wave Equation with Fixed Boundaries is defined as:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

The PINN was trained to simultaneously satisfy the initial conditions $u(x, 0)$ and $\partial u/\partial t$ at $t = 0$, boundary conditions $u = 0$ at $x = 0$ and $x = L$, and the differential equation itself via a residual term [14]. c represents the wave speed. A fully connected MLP was used, taking (x, t) as input and outputting $u(x, t)$. Parameters like layer count and size were configurable. An example comparison with an analytical solution is shown in Figure 1.

2D equation extends to two spatial dimensions (x, y):

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

This equation can describe, for example, membrane oscillations or other 2D wave processes. The same overall PINN scheme was used as in the 1D case, but network parameters were adjusted for increased complexity. A fully connected architecture was retained, with an increased layer size (256 neurons) to accommodate the 2D task's complexity. Optimizer details (Adam, learning rate, exponential decay) were specified.

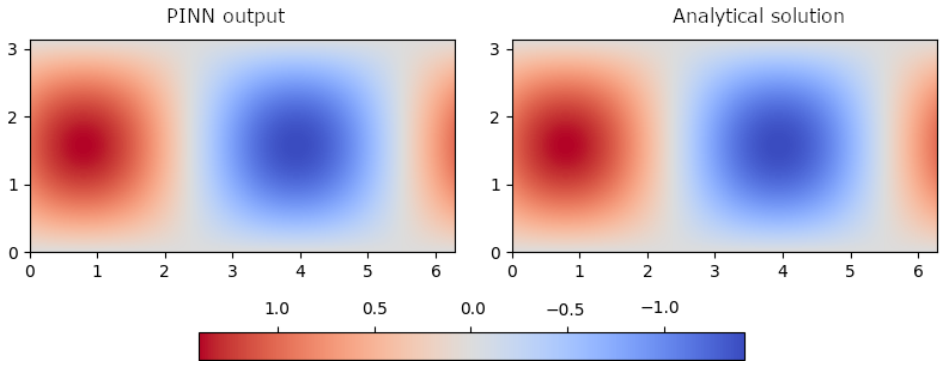


Figure 1. Comparison of PINN prediction with analytical solution.

Finally, 3D equation includes a third spatial dimension (z):

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

The simulation area was a cube. Initial conditions defined a harmonic function dependent on all three spatial coordinates, starting from rest. Zero displacement boundary conditions were applied on all six faces, simulating a fixed membrane. This configuration creates a complex interference pattern [12]. The same network architecture and hyperparameters as the 2D case were used for direct comparison of convergence and solution accuracy. A static cross-section visualizing the displacement field at a specific time slice showed solution symmetry and reflections from fixed walls, as seen in Figure 2.

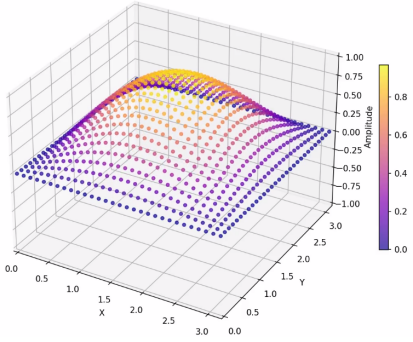


Figure 2. Visualization the three-dimensional wave equation.

Additionally, 2D wave equation with Perfectly Matched Layers (PML) and Obstacle was implemented. This complex case involved a 2D area with a circular obstacle and surrounding PML [28]. PML is a numerical technique to absorb

waves and simulate open boundaries, preventing unwanted reflections in wave process simulations like acoustic, electromagnetic, or elastic waves. It absorbs wave energy before it reaches the outer edge, simulating open space. The governing equations become a system involving the acoustic pressure and auxiliary fields to ensure absorption. A Neumann boundary condition $\partial u/\partial n = 0$ was applied on the obstacle's surface, simulating a perfectly rigid body reflecting waves [8]. Numerically, this condition is evaluated using components of the normal vector. The PINN formulation uses symbolic description of geometry, allowing precise application of boundary conditions on the curved surface by computing the normal using symbolic expressions [12]. An initial Gaussian pulse was defined. The network predicted the primary field (u), auxiliary PML fields (ψ_x, ψ_y), and additional quantities for normal derivatives. Due to the high complexity (multiple unknown functions, geometry, PML), optimizing the training process was necessary. An initial issue with GPU memory exceeding limits was resolved by halving the batch size, allowing training to complete with sufficient accuracy. Figure 3 shows the time evolution of the wave in this scenario.

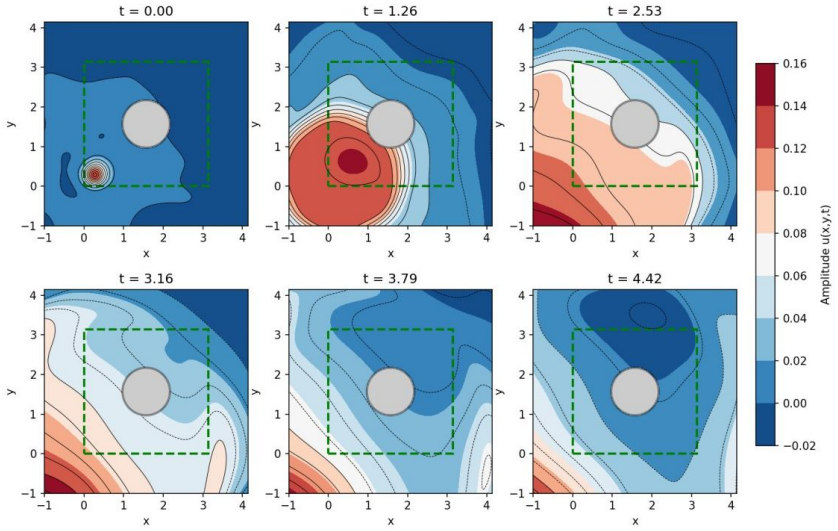


Figure 3. Time evolution of the acoustic wave in an area with an obstacle and absorbing PML.

Implementing these models in Modulus Sym involves defining the physics equations, specifying the geometry and boundary/initial conditions, configuring the neural network architecture and training parameters using YAML files, and setting up constraints and inferencers.

To model the underlying physical system, we employed a fully connected neural network using the Modulus Sym framework. The network architecture is designed to approximate the solution to a spatiotemporal problem by taking spatial coordinates (x, y) and time t as input variables, and producing a scalar output p,

representing the target physical quantity (e.g., pressure).

The network consists of five hidden layers, each containing 128 neurons. The activation function used across all layers is the hyperbolic tangent (*tanh*), which is commonly chosen for its smooth differentiability and favorable gradient properties when solving differential equations. The architecture is implemented using the FullyConnectedArch module in Modulus Sym, and is parameterized with appropriately defined symbolic input and output keys to facilitate automatic differentiation and integration with physics-informed loss functions.

To evaluate the predictive performance of the trained model, several error metrics were computed by comparing the predicted values p_{pred} with the reference data p_{ref} [22]. The element-wise difference between predicted and true values was first computed as $\Delta p = p_{\text{pred}} - p_{\text{ref}}$. Based on this difference, the following statistical indicators were calculated:

MSE: Computed as the average of the squared differences, it quantifies the mean magnitude of the prediction errors,

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\Delta p_i)^2.$$

RMSE: Defined as the square root of the MSE, providing an interpretable error magnitude in the same units as the output variable,

$$\text{RMSE} = \sqrt{\text{MSE}}.$$

Mean Error: The average of the raw differences between predicted and reference values,

$$ME = \frac{1}{N}\sum_{i=1}^{N}(p_{\Delta p_i}).$$

These metrics provide a comprehensive evaluation of model accuracy, consistency, and the nature of deviations from the true solution.

The quality of the obtained solutions was evaluated for all analyzed cases, from the basic 1D model to the most complex task with an obstacle and absorbing PML. The main objective was to verify the physical correctness of PINN predictions, compare results with reference solutions (analytical or numerical), and assess PINNs' ability to solve problems with complicated geometries and absorbing boundaries.

5. Results

Comparisons with analytical and FDM solutions for the 1D case showed very good agreement. Heat maps and wave profiles demonstrated minimal deviations. Quantitative metrics (MSE, L2-norm) were low (e.g., MSE of 1×10^{-6} vs. analytical, 6.7×10^{-5} vs. FDM for 1D; MSE of 5.437×10^{-7} vs. analytical, 5.347×10^{-7} vs. FDM for 2D; MSE of 6.256×10^{-6} vs. analytical, 6.253×10^{-6} vs. FDM for 3D). PINNs

successfully captured the dynamics of wave propagation with fixed boundaries in all dimensions. This confirmed PINNs' suitability for these academic examples and their ability to provide accurate solutions comparable to traditional methods like FDM. A key advantage highlighted was that PINNs do not require explicit mesh discretization [14]. In practical terms, the reported errors correspond to sub-percent deviations in wave amplitude, which are sufficient for engineering analysis where qualitative propagation patterns dominate. The accuracy achieved in 3D was sufficient for practical applications in complex configurations. As can be seen in the visualization in Figure 4, in the 3D wave case, the difference between the prediction and the analytical solution is minimal. In 1D and 2D cases, the error is so low that it can't be spotted visually. These results illustrate the robustness of PINNs for lower-dimensional problems and their potential as mesh-free solvers for higher dimensions, although the computational effort increases significantly compared to FDM. From the theoretical standpoint, the small residuals confirm that the physics loss term effectively constrained the solution across collocation points.

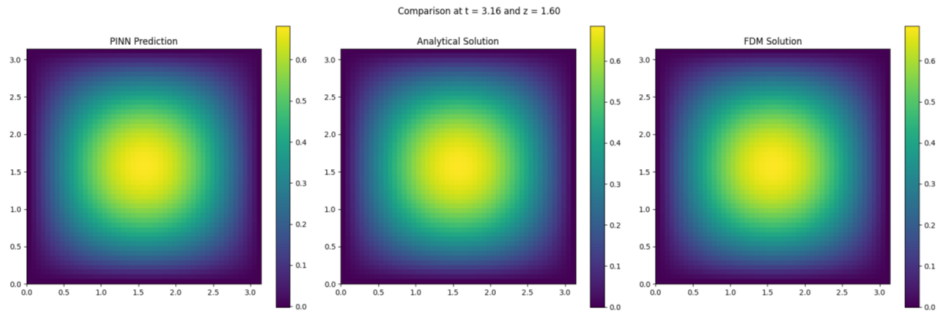


Figure 4. Comparison of FDM, PINN output and analytical solution for 3D wave at $z \approx 0$.

2D Acoustic Wave with Object and PML case was analyzed in parts: initial Gaussian pulse propagation, interaction with an obstacle, and absorption by PML.

Comparison of Gaussian Pulse in 2D without obstacle/PML with FDM showed acceptable accuracy. Visual differences were observed, particularly on the wave front. Quantitative metrics like relative L2 error were higher (8.2712×10^{-1}) compared to the simpler 1D/2D/3D cases. The relatively high error indicates that while PINNs can capture the main wave structure, sharp gradients and high-frequency components remain challenging, especially in higher-dimensional problems. Increasing network parameters improved accuracy but also slightly increased wave amplitude, indicating sensitivity to hyperparameters. Such behavior highlights a limitation of PINNs: their performance is strongly tied to architecture choices and may require extensive tuning, unlike FDM which has more predictable convergence behavior. Finding a closed-form analytical solution for a Gaussian pulse in 2D is challenging due to non-linearity and the need for complex functions like Bessel functions. The difficulty in this case reflects the challenge of minimizing $N(u_\theta(x_j))$

near steep gradients, where the residual term dominates the loss.

PINNs utilize a precise symbolic description of the circular obstacle geometry and boundary condition application [26]. Differences observed when comparing PINN results with FDM/FEM solutions are attributed primarily to the PINN's precise handling of the curved boundary condition via symbolic computation of the normal, contrasting with the approximate nature of this condition in FDM/FEM [3]. Such accuracy is particularly relevant in scenarios where faithful representation of curved or irregular boundaries is critical, such as biomedical or seismic wave simulations. Other sources of difference include numerical approximations in classical methods and potential discrepancies in initial impulse definition or time parameters. Aligning results requires more accurate boundary condition methods in classical techniques (e.g., immersed boundary, body-fitted mesh) and careful matching of initial and boundary conditions and discretization. A fundamental difference remains in how the solution is obtained: PINN optimizes a functional with analytical geometry, while FDM/FEM solve a discretized equation with precision limited by the discrete mesh [14]. The functional approach also enables PINNs to generalize solutions at arbitrary collocation points, whereas traditional methods are tied to predefined meshes. This observation directly illustrates the theoretical advantage of embedding the operator $N(u)$ in the loss: boundary conditions are enforced analytically rather than approximated numerically.

Comparison of PINN and FDM solutions of absorption in PML showed good overall approximation by PINN, but minor artifacts appeared in the upper region. These artifacts could potentially be due to imprecise PML boundary conditions in the PINN model. FDM showed clearer wave propagation. Quantitative errors; MSE=0.00321, RMSE=0.00545, with maximum error for a single collocation point 0.0213; indicated a small average deviation but significant local deviations on the wave front, suggesting PINNs' sensitivity to steep gradients. Such sensitivity limits their use in problems where sharp wavefront accuracy is critical (e.g., shock waves), unless specialized loss functions or adaptive sampling strategies are employed. The PML implementation in PINN showed artifacts, indicating it did not perfectly absorb waves, possibly due to suboptimal boundary condition setup in the loss function or architecture limitations. Further optimization was limited by computational resources. The growing training cost with domain complexity represents another limitation, which may offset the benefits of mesh-free formulation in large-scale 3D applications. From the perspective of the theorem, these artifacts indicate incomplete minimization of the residual at collocation points near the absorbing boundary, pointing to a need for refined loss weighting.

Three different neural network architectures were implemented for the 2D wave equation with PML and an object: Fully Connected, Fourier Neural Operator (FNO), and SIREN [3, 4, 9, 19]. Hyperparameters were kept the same across all cases for comparison, thus 5 layer network, 128 neurons per layer and tanh activation function for fully connected neural net and default Modulus values for FNO and SIREN.

The Fully Connected architecture yielded the best and most stable results

among those tested with the given configuration. It demonstrated smoother convergence and physically more correct wave behavior in the impulse problem. Classical dense architectures therefore remain a competitive baseline for wave problems, especially when training resources are constrained.

The FNO architecture showed a high initial loss that decreased rapidly in the first thousands of iterations, then settled at a lower level. This suggests FNO finds the main frequency components relatively quickly, but the solution was not completely ideal regarding reflection and absorption. Parameter tuning (normalization, number of harmonics) improved FNO results, reducing noise and yielding a recognizable impulse, but still not fully physically correct propagation, reflection, or absorption, see Figure 5. The gray circle represents an obstacle and the green square represents PML. These observations highlight both the promise of operator learning approaches for capturing global patterns, and their current limitations in faithfully reproducing fine-scale boundary interactions without tailored architectures.

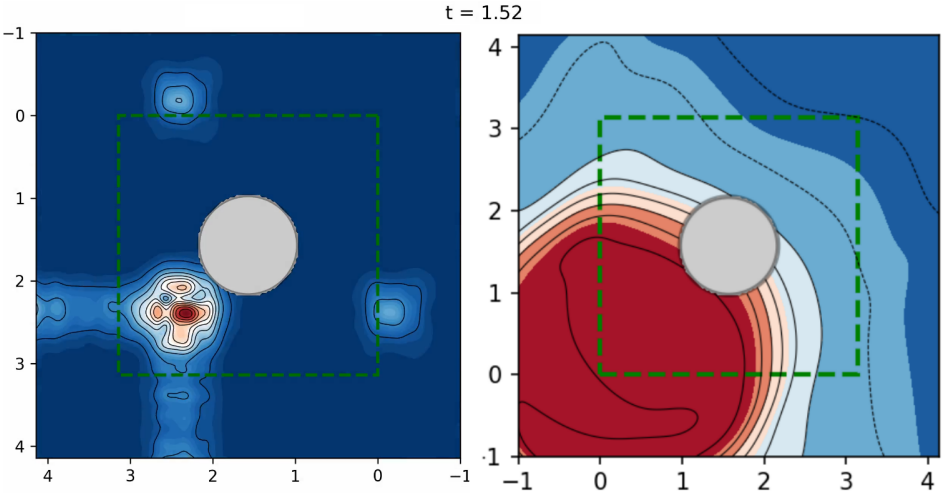


Figure 5. Result of the FNO model (left) after parameter optimization and Fully Connected (right) at the same time.

The SIREN architecture started with a very high loss and decreased slowly, remaining at a significantly higher value than the other architectures. The model struggled to find a sufficiently good representation of the wave field. For impulse disturbances with a wide frequency spectrum, SIREN typically requires special initialization techniques and frequency scaling. While SIRENs are powerful for high-frequency representations, their application to broadband wave propagation remains limited without problem-specific adaptations.

In summary, the results demonstrate that PINNs can achieve accuracy comparable to FDM and FEM in structured wave problems, with clear benefits in handling curved geometries and mesh-free generalization. However, their practical

deployment is limited by high computational cost, sensitivity to hyperparameters, and challenges with absorbing boundaries or sharp gradients. PINNs are therefore most promising for problems where geometric flexibility and smooth solutions are more critical than computational efficiency, such as biomedical or geophysical applications. Overall, the experiments confirm the theoretical formulation: minimizing the total loss $L(\theta) = L_{\text{data}} + L_{\text{physics}}$ produces solutions that not only fit data but also satisfy the governing PDE residuals within the domain.

6. Conclusion

This study investigated the application of PINNs for acoustic wave propagation prediction. A theoretical overview of acoustic waves and the PINN approach was provided, followed by simulations in NVIDIA Modulus Sym for one-, two-, and three-dimensional cases, including absorbing boundaries and rigid obstacles.

PINNs modeled wave phenomena in open boundaries and complex geometries without explicit space–time discretization, a key advantage over classical numerical methods. However, the models were sensitive to training parameters, especially in complex configurations (e.g., with impulse disturbances and PML), where artifacts, amplitude reduction, or wavefront deformation appeared. These issues arose from loss-function setup, excessive weighting of boundaries, uneven collocation points, or low weights in the interior domain.

Comparison with analytical, FDM, and FEM solutions confirmed the advantage of PINNs in applying boundary conditions to curved geometries via symbolic description. While accuracy was not uniform across the entire domain, PINNs generally preserved the physical structure of the solution and reproduced key wave properties.

Among Fully Connected, FNO, and SIREN architectures, the classical fully connected network yielded the most stable and physically correct results, emphasizing that architecture choice should reflect the disturbance type and expected solution form.

Although training is computationally intensive, PINNs' flexibility, mesh-free formulation, and incorporation of physical laws make them a promising tool for modeling complex physical processes. Future work should expand to more architectures, improve sampling strategies, and apply regularization to enhance accuracy, particularly for impulse waves and absorbing layers.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al.: *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, arXiv preprint arXiv:1603.04467 (2016).

[2] J. Brum: *Transverse wave propagation in bounded media*, Ultrasound elastography for biomedical applications and medicine (2018), pp. 90–104.

[3] S. Cai, Z. Mao, Z. Wang, M. Yin, G. E. Karniadakis: *Physics-informed neural networks (PINNs) for fluid mechanics: A review*, Acta Mechanica Sinica 37 (2021), pp. 1727–1738, doi: 10.1007/s10409-021-01148-1.

[4] K. Gurney: *An Introduction to Neural Networks*, Boca Raton, FL, USA: CRC Press, 2018.

[5] Y. Kagawa, T. Tsuchiya, T. Yamabuchi, H. Kawabe, T. Fujii: *Finite element simulation of non-linear sound wave propagation*, Journal of sound and vibration 154.1 (1992), pp. 125–145.

[6] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang: *Physics-informed machine learning*, Nature Reviews Physics 3 (2021), pp. 422–440, doi: 10.1038/s42254-021-00314-5.

[7] S. W. Kim, I. Kim, J. Lee, S. Lee: *Knowledge Integration into deep learning in dynamical systems: An overview and taxonomy*, Journal of Mechanical Science and Technology 35 (2021), pp. 1331–1342.

[8] L. E. Kinsler, A. R. Frey, A. B. Coppens, J. V. Sanders: *Fundamentals of Acoustics*, 4th ed., John Wiley & Sons, 2000.

[9] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar: *Fourier neural operator for parametric partial differential equations*, arXiv preprint arXiv:2010.08895 (2020).

[10] X. Luan, K. Yokota, G. Scavone: *Acoustic field reconstruction in tubes via physics-informed neural networks*, arXiv preprint arXiv:2505.12557 (2025).

[11] N. Mehtaj, S. Banerjee: *Scientific machine learning for guided wave and surface acoustic wave (SAW) propagation: PgNN, PeNN, PINN, and neural operator*, Sensors (Basel, Switzerland) 25.5 (2025), p. 1401.

[12] NVIDIA Corporation: *NVIDIA Modulus: Physics-Informed Neural Networks Framework*, https://developer.nvidia.com/modulus, 2024.

[13] A. Paszke: *Pytorch: An imperative style, high-performance deep learning library*, arXiv preprint arXiv:1912.01703 (2019).

[14] M. Raissi, P. Perdikaris, G. E. Karniadakis: *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational physics 378 (2019), pp. 686–707.

[15] M. Rasht-Behesht, C. Huber, K. Shukla, G. E. Karniadakis: *Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions*, Journal of Geophysical Research: Solid Earth 127.5 (2022), e2021JB023120.

[16] S. Schoder, F. Kraxberger: *Feasibility study on solving the Helmholtz equation in 3D with PINNs*, arXiv preprint arXiv:2403.06623 (2024).

[17] A. Sherstinsky: *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network*, Physica D: Nonlinear Phenomena 404 (2020), p. 132306, doi: 10.1016/j.physd.2019.132306.

[18] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, G. E. Karniadakis: *Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks*, Journal of Nondestructive Evaluation 39.3 (2020), p. 61.

[19] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein: *Implicit neural representations with periodic activation functions*, Advances in neural information processing systems 33 (2020), pp. 7462–7473.

[20] L. Sun, H. Gao, S. Pan, J. X. Wang: *Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data*, Computational Methods in Applied Mechanics and Engineering 361 (2020), p. 112732, doi: 10.1016/j.cma.2019.112732.

[21] H. Wang, J. Li, L. Wang, L. Liang, Z. Zeng, Y. Liu: *On acoustic fields of complex scatters based on physics-informed neural networks*, Ultrasonics 128 (2023), p. 106872.

[22] Q. Wang, Y. Ma, K. Zhao, Y. Tian: *A comprehensive survey of loss functions in machine learning*, Annals of Data Science 9.2 (2022), pp. 187–212.

[23] S. Wang, Y. Teng, P. Perdikaris: *Understanding and mitigating gradient flow pathologies in physics-informed neural networks*, SIAM Journal on Scientific Computing 43.5 (2021), https://arxiv.org/abs/2001.04536, A3055–A3081.

[24] H. Yamawaki, T. Saito: *Computer simulation of acoustic waves propagation in elastically anisotropic materials*, in: Materials Science Forum, vol. 210, Trans Tech Publ, 1996, pp. 589–596.

[25] L. Yang, X. Meng, G. E. Karniadakis: *B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data*, Journal of Computational Physics 425 (2021), p. 109913, doi: 10.1016/j.jcp.2020.109913.

[26] Y. Yang, P. Perdikaris, G. E. Karniadakis: *Physics-informed deep learning for the nonlinear dynamics of materials*, Computer Methods in Applied Mechanics and Engineering 389 (2022), p. 114378, doi: 10.1016/j.cma.2021.114378.

[27] K. Yokota, T. Kurahashi, M. Abe: *Physics-informed neural network for acoustic resonance analysis in a one-dimensional acoustic tube*, The Journal of the Acoustical Society of America 156.1 (2024), pp. 30–43.

[28] S. Yoshida: *Fundamentals of Acoustic Waves and Applications*, Springer, 2024.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 188–200
DOI: 10.17048/fmfai.2025.188

Benchmarking data logging strategies in ROS-integrated multi-sensor robots under network constraints

Nour Elhouda Ben Saadi[a], **Zoltán Istenes**[b]

Eötvös Loránd University
nourelhoudabensaadi@gmail.com
istenes@inf.elte.hu

Abstract. This work benchmarks multiple logging strategies, including JavaScript Object Notation (JSON) serialization, JSON Lines (JSONL), WebSocket based rosbridge, native Robot Operating System (ROS) Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) transports, and NFS-based recording, on a multisensor robot over a 100 Mbps constrained network. We evaluate dropped messages, resource usage, and storage footprint across high-throughput data streams, such as LiDAR, Red-Green-Blue plus Depth (RGB-D) images, Inertial Measurement Unit (IMU), and Global Navigation Satellite System (GNSS) data. Results show that TCPROS provides the highest reliability, while UDPROS, rosbridge, and JSON-based methods incur significant losses when network capacity is saturated. Network File System (NFS) recording performs well, provided that network stability is maintained. Our findings reveal key trade-offs between transport guarantees, network limitations, and message size fragmentation.

Keywords: ROS, data logging, benchmarking, TCPROS, UDPROS, JSON, multisensor robots, network constraints

AMS Subject Classification: 68T40, 68M10, 68M20

1. Introduction

Accurate and timely data logging in robotic systems is crucial for tasks such as performance evaluation, fault diagnosis, and developing machine learning pipelines. As robots increasingly integrate high-throughput sensors – such as Light Detection and Ranging (LiDAR), RGB-D cameras, inertial units, and GNSS modules – the abil-

ity to reliably record data is critical for both real-time decision-making and offline analysis. Recent advancements in ROS-based architectures, including modular sensor integration and networked communication protocols, have improved scalability and flexibility in robotic software systems. However, in bandwidth-constrained or infrastructure-limited environments – such as field robotics or industrial inspection – data transport can become a significant bottleneck. Under such conditions, large message sizes and unreliable links often cause dropped messages, increased Central Processing Unit (CPU) overhead, and inconsistent logging performance. Efforts like ROS Enhancement Proposal (REP)2014 propose benchmarking guidelines for ROS 2 systems to standardize performance analysis under diverse computational and network conditions [5]. The RobotPerf suite [4] further enables reproducible benchmarking of computational workloads in ROS 2 pipelines. Additionally, recent studies on ROS 2 real-time latency and transport performance [2, 12] demonstrate the value of systematic evaluations across middleware configurations. However, despite ROS 2 gaining popularity, ROS 1 remains widely deployed across academic and industrial settings, especially in long-lived platforms and legacy deployments. While most recent benchmarks focus on ROS 2, there is a lack of in-depth evaluation of logging strategies in ROS 1, particularly under constrained network conditions, where transport choice, message encoding, and system load directly affect data reliability. To address this gap, we conduct a comprehensive benchmarking study of ROS 1 data logging strategies in multi-sensor robotic systems, focusing on real-world limitations caused by limited bandwidth.

Our key contributions are as follows:

- We benchmark six data logging strategies in ROS 1: TCPROS, UDPROS, rosbridge (WebSocket), JSON serialization, JSONL, and NFS-based remote logging.
- We evaluate each strategy in a real-world multi-sensor setup (LiDAR, RGB-D, IMU, GNSS) across a 100 Mbps constrained network.
- We compare performance based on message loss, CPU and memory usage, and storage efficiency.
- We analyze trade-offs related to transport guarantees, serialization overhead, and system saturation under network stress.

In this paper, we provide a comprehensive evaluation of data logging strategies in ROS 1 under constrained network conditions on a multi-sensor robotic platform. Unlike prior studies, our benchmarks are performed in real time during active sensor streaming, capturing the actual performance limitations encountered in deployment scenarios. The remainder of this paper is structured as follows: Section 2 discusses related work. Section 3 presents the system architecture and logging strategies. Section 4 details the benchmarking methodology. Section 5 presents the results and their analysis. Finally, Section 6 concludes and outlines future directions.

2. Related work

Benchmarking in robotic systems has gained increased attention with the evolution of modular and distributed architectures, particularly in ROS 2. The official benchmarking guideline, REP-2014, provides recommendations for reproducible performance analysis in ROS 2 systems, including tracing, latency measurement, and profiling [5]. Building on this, the RobotPerf suite offers a vendor-neutral benchmarking framework for computational performance in ROS 2, targeting CPU usage, scheduling behavior, and system latency under sensor load [4]. However, both focus on ROS 2 middleware and compute pipelines rather than message transport or data logging behavior. Several other works have examined ROS 2 communication performance. Kronauer et al. analyzed latency behavior in multi-node Data Distribution Service (DDS) based systems and highlighted the trade-offs between configuration settings and timing guarantees [2]. Yanlei et al. conducted a comparative evaluation of ROS 1 and ROS 2 real-time performance under CPU load, noting ROS 2's architectural improvements in timing determinism [12]. Still, these studies do not explore logging reliability or network-induced message loss. In the ROS 1 ecosystem, tools like ROS-CBT focus on communication benchmarking by measuring throughput and latency over emulated network links between virtual nodes [3]. While ROS-CBT includes tests under constrained bandwidth, it does not evaluate end-to-end logging mechanisms or analyze serialization overhead, system load, or message loss during real-time recording. Furthermore, it is simulation-based and does not reflect the challenges of logging live, high-throughput sensor data on deployed robotic systems. Other works like ROSfs propose user-space file systems for sharing ROS data across distributed robots, but they do not benchmark performance or logging efficiency under network constraints [11]. To the best of our knowledge, no prior work has systematically evaluated data logging strategies in ROS 1 under constrained network conditions using a live, multi-sensor robotic platform. Our study fills this gap by benchmarking six strategies – including TCPROS, UDPROS, and NFS recording, as well as rosbridge-based logging of ROS messages in raw ROS Bag file format for logging messages (rosbag) format, JSON, and JSONL. We evaluate message loss, CPU and memory usage, and storage footprint under real-time conditions with active sensor streaming.

3. System description

Our evaluation was conducted on a ROS 1-based robotic platform – we named Pomona – designed for sensor-rich navigation and mapping tasks. The robot we used is an Agile-X four-wheel-drive Scout Mini[1] robot equipped with the Research Pro sensor kit (see Figure 1).

The system integrates a mix of high- and low-throughput sensors, covering a broad range of data rates including:

[1] https://global.agilex.ai/products/scout-mini

Figure 1. Pomona: a ROS 1-based multi-sensor robotic platform used in our experiments. The system integrates a Velodyne VLP-16 LiDAR, an Intel RealSense D435 RGB-D camera, an Xsens IMU, and a GNSS receiver.

- **LiDAR**: Velodyne VLP-16 (`/velodyne_points`, `sensor_msgs/PointCloud2`), 10 Hz, average bandwidth ~6.59 MB/s, average message size ~0.62 MB
- **RGB-D camera**: Intel RealSense D435
  - **Color image** (`/camera/color/image_raw`, `sensor_msgs/Image`): 30 Hz, average size ~0.92 MB, bandwidth ~27–28 MB/s
  - **Depth image** (`/camera/depth/image_raw`, `sensor_msgs/Image`): 30 Hz, average size ~0.81 MB, bandwidth ~25 MB/s
  - **Compressed color** (`/camera/color/image_compressed`, `sensor_msgs/CompressedImage`): 30 Hz, average size ~16.5 KB, bandwidth ~510 KB/s
  - **Compressed depth** (`/camera/depth/image_compressed`, `sensor_msgs/CompressedImage`): 30 Hz, average size ~22 KB, bandwidth ~670 KB/s
- **IMU**: Xsens MTI 600 series (`/imu/data`, `sensor_msgs/Imu`), 25 Hz, message size ~0.32 KB, bandwidth ~8 KB/s
- **GNSS receiver**[2]: (`/gnss`, `sensor_msgs/NavSatFix`), 4 Hz, message size ~124 bytes, bandwidth ~640 B/s
- **Base controller**: Scout Mini (`/scout_status`, `scout_msgs/ScoutStatus` [1]), 50 Hz, message size ~0.14 KB, bandwidth ~7.3 KB/s

[2]Incorporated into the Xsens IMU

These sensors interface with an onboard NVIDIA Jetson AGX Xavier running ROS Melodic (Ubuntu 18.04), streaming data over a 100 Mbps Ethernet link to an external logging workstation (Ubuntu 22.04). This constrained bandwidth emulates real-world field robotics and industrial inspection scenarios, where infrastructure or power limitations restrict data transmission. Under these conditions, we evaluated the following six logging strategies.

3.1. TCPROS (TCP)

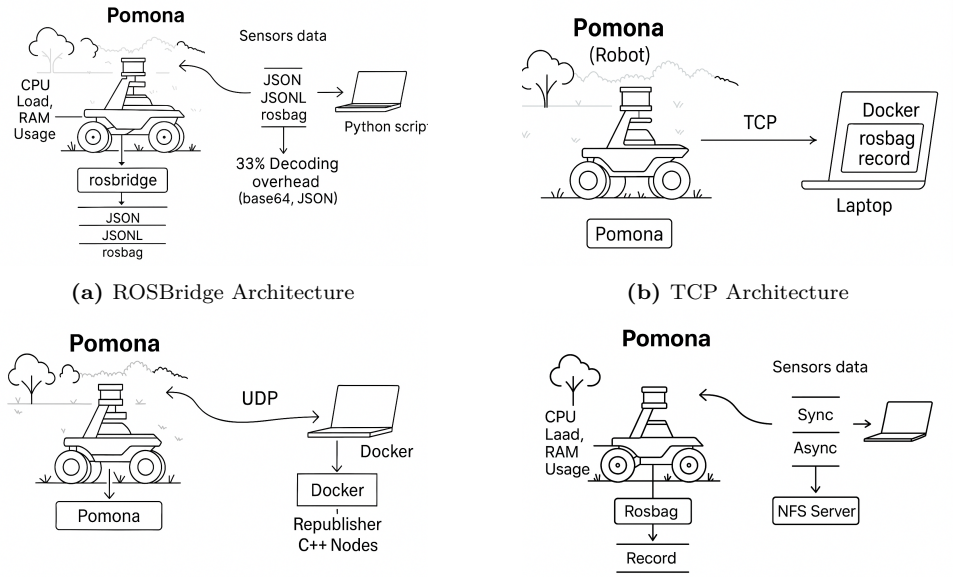
TCPROS [6] is the default ROS 1 transport, using TCP for reliable, in-order delivery between publishers and subscribers. As illustrated in Figure 2b, Pomona's sensor nodes publish topics over the 100 Mbps Ethernet link to a *Docker* container running on the logging workstation. The container uses a ROS 1 image (e.g., `ros:melodic-ros-base`) to provide the correct environment for `rosbag record`. This containerized setup is necessary because the logging workstation runs Ubuntu 22.04, which is incompatible with ROS Melodic, and it avoids installation conflicts or dependency issues on the host system [9, 10]. Host networking (`-network host`) and a bind mount to `/data` enable direct topic subscription and efficient data storage. Each publisher–subscriber pair establishes a persistent TCP channel. TCP handles retransmissions and enforces ordering, ensuring message integrity for high-bandwidth topics such as LiDAR scans, RGB-D images, and IMU readings. Under high network utilization, TCP back-pressure increases ROS publish queues, and if these queues overflow, messages are dropped at the publisher before transmission.

3.2. UDPROS (UDP)

UDPROS [7] is a connectionless ROS 1 transport that uses the User Datagram Protocol (UDP) to transmit messages without delivery guarantees. As illustrated in Figure 2c, Pomona's sensor nodes publish topics over the 100 Mbps Ethernet link to a *Docker* container on the logging workstation running a ROS 1 Melodic image. Direct use of `rosbag record` could not successfully negotiate the UDP transport layer with Pomona's publishers, as ROS 1 defaults to TCPROS unless both endpoints explicitly advertise UDPROS support. To enable UDP-based recording, we implemented an intermediate C++ relay node inside the container. This node subscribed to Pomona's topics via UDPROS, confirmed using `rostopic info`, and re-published the messages locally within the container. `rosbag record` then subscribed to these local topics, benefiting from intra-process communication while preserving the UDP characteristics of the original inbound link.

3.3. rosbridge + rosbag

The `rosbridge_server` [8] exposes ROS topics over a JSON/WebSocket interface. As illustrated in Figure 2a, Pomona runs `rosbridge_server` and streams messages to the logging workstation over WebSocket. On the workstation, we



(a) ROSBridge Architecture

(b) TCP Architecture

(c) UDP Architecture

(d) NFS Architecture

Figure 2. Comparison of architectures: ROSBridge, TCP, UDP, and NFS.

run a *Docker* container with a ROS 1 Melodic image to provide the required environment for `rospy`, `rosbag`, `genpy`, and message definitions. Inside the container, a Python bridge (via `roslibpy`) subscribes to the WebSocket topics, reconstructs native ROS messages, and writes them to a `.bag`. For message types with binary payloads (`sensor_msgs/CompressedImage`, `sensor_msgs/PointCloud2`), `rosbridge_server` transmits the `data` field as base64 text; we therefore *decode to raw bytes prior to message construction*. Without this step, the bag appears valid in `rosbag info` but the payload bytes are incorrect, leading to corrupted images and point clouds at playback.

3.4. rosbridge + JSON

In this configuration (Figure 2a), Pomona's onboard ROS 1 Melodic system runs `rosbridge_server` to serialize ROS messages into JSON and stream them over a WebSocket connection. The logging workstation connects via `roslibpy` from a Python script and saves incoming messages directly to a JSON file for each topic. The subscriber script maintains an in-memory dictionary keyed by topic name and appends each received message before periodically writing to disk. For topics containing binary payloads (e.g., `sensor_msgs/Image`, `sensor_msgs/CompressedImage`, `sensor_msgs/PointCloud2`), the raw JSON representation can produce very

large files and, if not explicitly base64-encoded before transmission, may risk truncation or corruption depending on client library handling. This approach bypasses `rosbag record` entirely, allowing collection on systems without a full ROS environment, but requires careful handling of binary fields to ensure data integrity.

3.5. rosbridge + JSONL

In this configuration (Figure 2a), Pomona runs `rosbridge_server` to serialize ROS messages into JSON objects, which are transmitted over a WebSocket connection to the logging workstation. Instead of aggregating messages into a single file, each incoming message is written as an individual JSON object on its own line (JSON Lines, or JSONL format). The subscriber script, implemented in Python with `roslibpy`, appends each received message to a text file as soon as it arrives. This streaming write pattern enables incremental storage without keeping the entire dataset in memory and simplifies parsing for large-scale post-processing, since each line is a self-contained JSON object. As with the pure JSON approach, topics containing binary fields (e.g., `Image`, `CompressedImage`, `PointCloud2`) require base64-encoding of their `data` fields to avoid payload corruption. Without decoding these fields back into raw bytes during reconstruction, the resulting files will appear structurally valid but contain unusable image or point cloud data.

3.6. NFS recording

In this configuration (Figure 2d), the logging workstation mounts a remote directory from Pomona via the Network File System (NFS) protocol. On Pomona, `rosbag record` runs natively within the ROS 1 Melodic environment, writing directly to the mounted path, which physically resides on the workstation's storage. This arrangement keeps all ROS topic subscription, serialization, and bag creation on the robot side, while the workstation receives the bag file writes in real time through the NFS link. It avoids the need for a ROS environment on the workstation for recording and guarantees full compatibility with Pomona's Melodic setup. However, overall performance is bound by NFS throughput and the robot's disk I/O; high-rate or large-payload topics may still cause write delays or dropped messages if the network or storage becomes saturated.

4. Benchmarking methodology

All strategies were tested in real time while the robot continuously published sensor data. Logging scripts were implemented in Python and C++ where applicable. Depending on the strategy, performance metrics were collected in the robot or in the external logging machine. These metrics include CPU load, memory usage, storage footprint, and message drop rate. While Linux tools such as `htop`, `iftop` were consulted for runtime inspection, we primarily used `time -v` to obtain detailed per-process performance metrics at critical evaluation points during logging.

To evaluate the performance of each logging strategy under bandwidth- and resource-constrained conditions, we conducted real-time experiments with the robot Pomona while it actively streamed high-rate sensor data. Each strategy was launched independently while keeping the sensor configuration constant to ensure comparability. For each strategy, we performed at least three independent runs, each lasting 120 seconds, to capture representative performance and account for run-to-run variability. Performance data were recorded concurrently on either the onboard or the external logger, depending on where the logging process was executed. Metrics measured included:

- **CPU usage**: peak and average utilization of the logging process, using: `time -v`
- **Memory consumption**: maximum Resident Set Size (RSS), from `time -v`.
- **Bandwidth utilization**: observed using `iftop` on the robot's Ethernet interface (note: not measured separately for each logging strategy).
- **Storage footprint**: file size of logged data.
- **Message drop rate**: estimated based on comparing the expected message counts, computed as:

$$\text{Expected Count} = f_{nominal} \times T_{recording} \tag{4.1}$$

  where $f_{nominal}$ is the nominal topic frequency and $T_{recording}$ is the logging duration, with the actual received counts in the logs. The drop percentage was then calculated as:

$$\text{Drop Rate (\%)} = \frac{\text{Expected Count} - \text{Actual Count}}{\text{Expected Count}} \times 100 \tag{4.2}$$

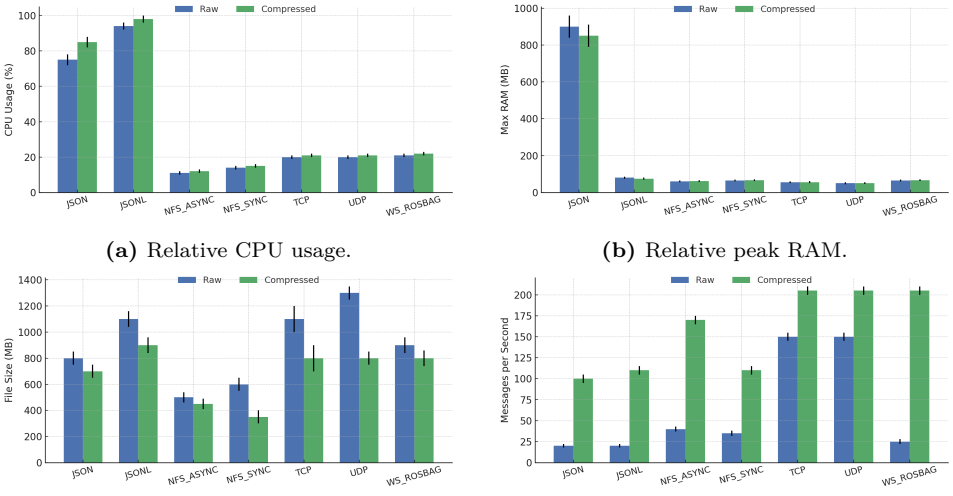
  This method allowed us to estimate the proportion of received messages relative to the expected volume, providing a quantitative basis for drop rate analysis.

The `time -v` outputs were saved automatically for each run and processed using a custom script to generate tabular data, which were then read into Pandas data frames for plotting and further analysis. For fairness, background system load was kept minimal during all runs. For both persistent-mode experiments and standard runs that completed successfully without anomalies, all reported metrics are presented as the mean together with the standard deviation, computed across multiple experimental repetitions.

5. Results and analysis

5.1. System performance and throughput

JSON vs JSONL. Figure 3 compares key performance metrics for all logging strategies under raw and compressed configurations. CPU usage (Figure 3a) shows



(a) Relative CPU usage.

(b) Relative peak RAM.

(c) Relative output file size.

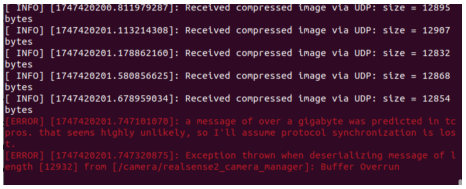
(d) Relative message throughput.

Figure 3. Relative system load and throughput metrics for raw and compressed configurations across all methods.

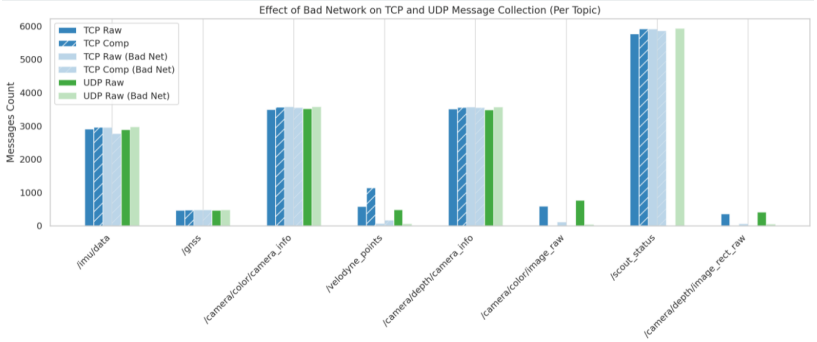
that JSONL, while still exhibiting high CPU load due to its intensive disk I/O, maintains the low RAM usage (Figure 3b). This allows continuous recording without memory saturation, in contrast to the JSON strategy where unsaved data accumulates in RAM until a flush to disk is triggered. Under long runs, this progressive growth (shown in Figure 6a) continues until the output file size exceeds host-level limits, at which point logging halts with an `Errno 27: File too large` error and incomplete data persistence (Figure 4a). To further investigate JSONL's behavior, we experimented with tuning its buffering size (Figure 6c). While smaller buffers kept message throughput relatively stable over extended runs, larger buffers initially increased the sustained rate but eventually led to a sudden drop after prolonged operation. This effect is attributed to accumulated write delays and internal queue growth, which cause bursts of backpressure when the disk subsystem is momentarily saturated. The behavior, although less severe than JSON's abrupt halts, still indicates a limit to how long JSONL can maintain peak throughput without periodic flushing or additional I/O optimization.

Native ROS transports (TCPROS, UDPROS) and NFS-based logging maintain low CPU and RAM usage, with minimal differences between raw and compressed operation. File size results (Figure 3c) highlight the trade-off between storage footprint and message throughput (Figure 3d). Compression reduces output size across all methods, but throughput generally drops for more serialization-heavy approaches. A notable exception is the `ws_rosbridge_rosbag` method, which performs poorly on raw data but shows throughput in the compressed configuration comparable to TCPROS and UDPROS. This counter-intuitive result – achieving

(a) JSON method halting due to file-size limitation (`Errno 27`).



(b) UDP logging failure during compressed image streaming (`Buffer Overrun`).



(c) Network instability affecting both TCP and UDP transports, causing simultaneous camera and LiDAR dropouts until Wi-Fi reset.

Figure 4. Observed runtime failures and instability. Top: method-specific logging errors (a: JSON, b: UDP). Bottom: link-level instability affecting native ROS transports (c).

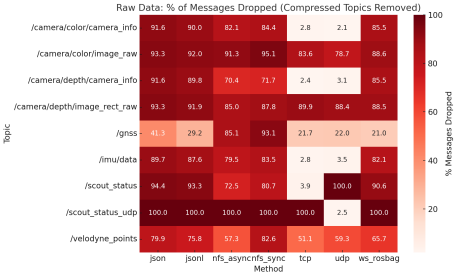
similar throughput while producing smaller files – suggests that compression mitigates bottlenecks in the WebSocket and `rosbag` pipeline, allowing data to be processed and written at a higher sustained rate.

UDP instability and MTU sensitivity. Beyond the JSON limitations, UDP exhibited run-time instability that forced a hybrid setup. When large payloads (camera frames/LiDAR packets) approached or slightly exceeded the link MTU, the UDPROS stream intermittently stalled. Lost fragments caused transport desynchronization, manifesting as spurious size predictions and `Buffer Overrun` errors in the camera node (Figure 4b). Increasing socket and ROS-level buffers reduced drops for small and medium topics, but did not prevent stalls for large payloads under link instability. During affected runs we restored operation by republishing locally (UDP relay → local topics). As a practical mitigation, lowering image resolution/quality or enabling camera-side compression reduces packetization pressure; otherwise TCPROS is preferable for large messages on unstable links. Given this fragility, the Reliability results in Section 5.2 correspond to a *hybrid transport configuration*: UDPROS was retained for low-bandwidth, latency-sensitive topics (IMU, GNSS, status), while large-payload topics (camera frames, LiDAR scans)

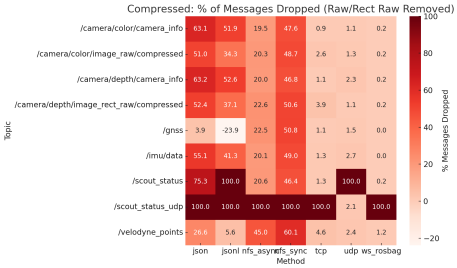
were switched to TCPROS to prevent fragmentation stalls and ensure reliable delivery under variable link conditions. As shown in Figure 6b, this hybrid approach preserved the high delivery rate for small topics while restoring near-complete reception for large payloads.

General network instability. In addition to protocol-specific issues, several experimental runs in both TCP and UDP experiments, intermittent Wi-Fi link throttling impacted performance, where throughput dropped sharply even though the link did not fully disconnect. This manifested as sudden throughput drops and stalled topics certainly over heavy connections, and in some cases required manually resetting the wireless interface to recover (Figure 4c). While rare, such events highlight the importance of robust reconnection and buffering logic in real-world field deployments.

5.2. Reliability: Message Delivery

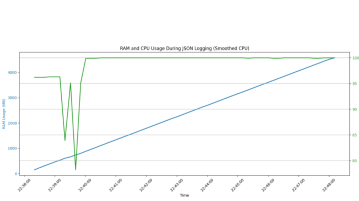


(a) Raw data. Compressed topics omitted.

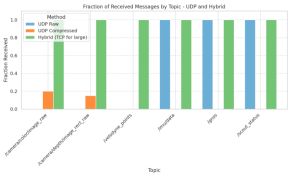


(b) Compressed data. Raw image topics omitted.

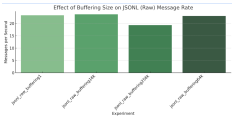
Figure 5. Percentage of messages *dropped* per topic and method for raw vs. compressed data streams.



(a) RAM and CPU usage during JSON logging.



(b) Fraction of received messages for UDP raw vs hybrid transport.



(c) Effect of buffering size on JSONL (raw) message rate.

Figure 6. Additional runtime observations: (a) CPU and RAM growth in JSON logging; (b) improvement in message delivery using hybrid UDP/TCP; (c) impact of buffer size on JSONL throughput.

Drop rates. Figures 5a–5b report message loss as Equation 4.2 indicates. On

raw streams (Figure 5a), JSON/JSONL exhibit consistently high loss across topics, while native transports (TCPROS, UDPROS) retain most control/metadata messages (e.g., `camera_info`, `imu/data`) but still struggle with raw image topics. The dedicated UDP status topic (`/scout_status_udp`) is correctly preserved under UDPROS, matching the transport design. With compression enabled (Figure 5b), TCPROS, UDPROS, and `ws_rosbag` achieve near-zero loss across the board, whereas JSON/JSONL still drop substantially on high-rate sensors. We also observe occasional $> 100\%$ "received" artifacts (visible as negative drop in the source percentage plots) on JSONL channel corresponding to the `/gnss` topic, suggesting message duplication on the WebSocket path rather than true reliability gains. Overall, compression shifts the balance strongly in favor of native ROS transports and the `ws_rosbag` pipeline for dependable delivery under the constrained link.

6. Conclusion and future work

This study provided a systematic benchmarking of ROS 1 data logging strategies under varying payload sizes, compression settings, and network conditions. By evaluating CPU usage, memory footprint, storage requirements, and message delivery rates, we mapped the trade-offs that influence transport and storage performance in real-world deployments. Throughput-oriented transports such as TCPROS, UDPROS, and `ws_rosbag` achieved the highest message rates, particularly with compression enabled. However, UDP's strong performance on small, latency-sensitive topics was offset by its fragility under network stress, making hybrid strategies – where large payloads fall back to TCP – more robust. NFS-based logging, especially asynchronous NFS, offered excellent CPU efficiency but struggled with sustained high-frequency data streams. WebSocket-based JSON/JSONL methods – while easy to set up and flexible – incurred heavy CPU load and high drop rates on large data streams due to serialization overhead. Across all methods, raw (uncompressed) configurations were more vulnerable to packet loss when network capacity was saturated.

No single method proved universally optimal across all conditions tested: the best choice depends on application constraints such as CPU availability, network stability, required throughput, and tolerance for packet loss. Our results provide empirically grounded guidance for selecting or combining transports to meet specific operational requirements in ROS 1-based robotic systems.

Future work will integrate precise time synchronization (e.g., NTP/chrony or hardware timestamping) to enable accurate end-to-end latency analysis. We also plan to extend this benchmarking framework into a more automated, deployment-agnostic tool that can isolate bottlenecks, profile shared resources, and adapt transport selection in real time based on topic characteristics and network state. Additional research will address challenging domains such as real-time LiDAR compression, power and thermal budgeting for edge deployments, validation in mobile, unstable wireless environments, and a hybrid migration strategy in which the work-

station transitions to DDS-based ROS 2 for improved resilience, while the robot remains on ROS 1 and communicates via rosbridge for interoperability. These steps will move toward adaptive logging systems that dynamically balance throughput, resource usage, and reliability in diverse field robotics applications.

References

[1] AgileX Robotics: *Scout_msgs ROS Package*, Accessed: 2025-08-10, 2025, URL: https://github.com/agilexrobotics/scout_ros/tree/master/scout_msgs/msg.

[2] T. Kronauer, J. Pohlmann, M. Matthe, T. Smejkal, G. Fettweis: *Latency Analysis of ROS2 Multi-Node Systems*, arXiv preprint arXiv:2101.02074 (2021), https://arxiv.org/abs/2101.02074.

[3] K. Masaba, A. Quattrini Li: *ROS-CBT: Communication Benchmarking Tool for the Robot Operating System: Extended Abstract*, in: Aug. 2019, pp. 1–3, DOI: 10.1109/MRS.2019.8901094.

[4] V. Mayoral-Vilches, J. J. Jabbour, Y.-S. Hsiao, et al.: *RobotPerf: An Open-Source, Vendor-Agnostic Benchmarking Suite for Evaluating Robotics Computing System Performance*, arXiv preprint arXiv:2309.09212 (2023), https://arxiv.org/abs/2309.09212.

[5] V. Mayoral-Vilches, I. Lütkebohle, C. Bédard, R. Araújo: *REP-2014: Benchmarking performance in ROS 2*, ROS Enhancement Proposal, https://ros.org/reps/rep-2014.html, 2022.

[6] Open Robotics: *ROS TCPROS Transport*, Accessed: 2025-08-10, 2025, URL: http://wiki.ros.org/ROS/TCPROS.

[7] Open Robotics: *ROS UDPROS Transport*, Accessed: 2025-08-10, 2025, URL: http://wiki.ros.org/ROS/UDPROS.

[8] Open Robotics: *rosbridge Suite*, Accessed: 2025-08-10, 2025, URL: http://wiki.ros.org/rosbridge_suite.

[9] O. Robotics: *ROS Distributions and Ubuntu Compatibility*, Accessed: 9 August 2025, 2025, URL: https://wiki.ros.org/Distributions.

[10] O. Robotics: *ROS Melodic Morenia*, Accessed: 9 August 2025, 2023, URL: https://wiki.ros.org/melodic.

[11] S. Schneegass, P. Weiss: *ROSfs: A File System for Robot Operating System Based Data Exchange*, arXiv preprint arXiv:2406.10635 (2024), https://arxiv.org/abs/2406.10635.

[12] Y. Yanlei, Z. Nie, X. Liu, F. Xie, Z. Li, P. Li: *ROS2 Real-time Performance Optimization and Evaluation*, Chinese Journal of Mechanical Engineering 36.1 (2023), p. 144, DOI: 10.1186/s10033-023-00976-5.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 201–213
DOI: 10.17048/fmfai.2025.201

Automata-based representation of coordination for distributed reactive systems*

Richárd Szabó, Dóra Cziborová

Budapest University of Technology and Economics
Department of Artificial Intelligence and Systems Engineering
Budapest, Hungary
{szabor,cziborova}@mit.bme.hu

Abstract. Modern cyber-physical systems (CPS) are distributed reactive real-time systems used in many critical application domains, such as automotive or railway systems, so ensuring their correctness is essential. Formal verification can exhaustively explore the behavior of the formal representation of CPS to ensure its reliability. Engineering modeling tools provide separate modeling constructs for the different aspects of the systems, e.g., behavior, architecture, and scheduling, but lack formal composition, making system-level verification difficult. Formal modeling tools, e.g., Lingua Franca or the Gamma Statechart Composition Framework, are efficient in describing component behavior, and they provide formal composition semantics of the subsystems. However, the provided composition patterns in these languages are not general, so incorporating the precise coordination, scheduling, and interaction aspects requires the modification of the component models by encoding the coordination into the components. In this paper, we present a configurable formal description approach for the coordination of distributed critical systems. We extend the well-known timed automata formalism to coordinate the execution of the components by directly reusing the formal models of the components, making the coordination of the system components a first-class citizen.

Keywords: formalization, coordination, distributed systems, system modeling

*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.

1. Introduction

The modeling and verification of modern cyber-physical systems (CPS) present several unique challenges that arise due to the integration of distributed heterogeneous components. These systems integrate physical systems with HW/SW components across varying platforms and locations. One challenge is to account for the timing of distributed components, which can be influenced by the communication network between the distributed components (messages can be delayed or lost). CPS are often used in critical application domains, e.g., the railway or the automotive industry. One way to ensure the reliability and correctness of these systems is to apply formal verification techniques like model checking, which systematically explores the state space of the system.

Various modeling and formal languages are developed to help engineers use formal verification: engineering modeling tools, such as AUTOSAR, aim for efficient engineering and provide separate modeling constructs for the different aspects of the systems, e.g., behavior, architecture, and scheduling. However, no formal composition is defined, so it is hard to target the system-level behavior with formal verification. On the other hand, formal modeling tools, e.g., Lingua Franca or the Gamma Statechart Composition Framework, are efficient in describing component behavior and they provide formal composition semantics of the subsystems. However, the provided composition patterns in these languages are not general, so incorporating the precise coordination, scheduling, and interaction aspects in these tools requires the modification of the component models and the encoding of the coordination into the components. Additionally, in distributed CPS, the underlying formal models of the communication and the subsystems often vary based on the application domain, network architecture, and system requirements, thus, a configurable solution is needed to model the coordination of the system.

In this paper, we present a configurable formal description approach for the coordination of distributed critical systems. We extend the well-known timed automata formalism to coordinate the execution of the components by directly reusing the formal models of the components, making the coordination of the system components a first-class citizen.

1.1. Motivating example: Steer-by-Wire

In our previous works, we presented an approach to model time-dependent behaviors in complex distributed systems and showed its applicability with a Steer-by-Wire (SbW) system inspired by our industrial partner [5], and we investigated how the coordination of the SbW system can be modeled [10].

In a SbW system to provide steering functionality, the road wheels are actuated via a control loop. Unlike classic Electric Power Assisted Steering (EPAS) systems, in the case of an SbW system, there are no direct mechanical connections between the steering wheel and the road wheels. The angle of the steering wheel is measured by multiple sensors, and the road wheels are actuated by the *Road Wheel Actuator*

(RWA) subsystems based on the measurements. Since actuating the road wheel is a critical function, a Master Selection Protocol (MSP) was developed to control the actuation of the road wheels. The MSP assigns which of the redundant RWAs must be used to control the road wheels based on the availability and correctness of their sensor measurements. In the presented case study, there are two RWAs and four sensors, as depicted in Figure 1.

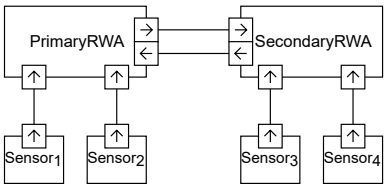


Figure 1. Architecture of the SbW system.

The components of the SbW system are distributed: the components are deployed on separate *electrical/electronic* (E/E) subsystems, and the components communicate over redundant *Controller Area Network* (CAN) buses and private communication buses. The execution of the E/E subsystems is triggered by their clocks, and the clocks can deviate from each other. Furthermore, the CAN buses are used by other components of the system. The combination of these properties can cause delayed messages; in extreme cases, this can lead to comparing sensor measurements originating from different execution cycles, possibly deeming correct measurements as faulty. Selecting both RWAs as master or selecting a failed RWA as the master can lead to accidents. To verify that the deviations of the E/E subsystem clocks cannot cause erroneous master selection, a sufficient formal representation of the coordination of the subsystems is needed. In order to formally model and verify such a complex system, we need a configurable modeling language that allows us to describe the possible execution scenarios, and while the previous works presented in Subsection 2.1 all have their strength, they lack the option to model the possible execution scenarios as first-class citizens.

2. Background and related work

2.1. Modeling complex distributed systems

There are several widely used modeling languages for defining the architecture and behavior of a system. These languages have varying levels of precision in their semantics. To use mathematical tools for the systematic examination of a system's design, a formal model of the system with mathematically precise semantics is needed [3].

Lingua Franca [7] is a coordination language to model concurrent reactive components (reactors). Lingua Franca works with both logical time (discrete ordering of events) and physical time (timestamps). The goal of Lingua Franca is to provide a deterministic model of the system by utilizing the priorities of the reactors, the dependencies between the reactors, and the logical and physical timing of events. Even though the goal of the authors of Lingua Franca is to model deterministic systems, nondeterminism is allowed if explicitly required by the engineer.

The *Gamma Statechart Composition Framework* [8] focuses on bridging the gap between higher-level (engineering) models and lower-level (formal) models and facilitates the modeling and verification of component-based reactive systems. The framework provides semantic-preserving model transformations between higher-level and lower-level modeling languages. The framework supports the hierarchical composition of components, which enables the engineers (1) to break down the system into smaller, reusable subsystems or (2) connect distributed components to provide higher-level system functionalities.

Similarly to the coordination automata formalism presented in Subsection 3.1, Norström et al. [9] present an extension of the timed automata. Their formalism, the *task automata*, enables the schedulability analysis of tasks by performing reachability analysis. In [1], the authors present a timed labeled transition system with invariants to define controlling and priorities of applications. The requirements of the system are mapped to specific transitions, thus requiring a white-box model of the system. Our proposed formalism considers the coordinated subsystems as black-box components since the transitions of the subsystems are not modified.

2.2. The TXSTS modeling formalism

The *timed extended symbolic transition system* (TXSTS) formalism (proposed in [4] as an extension of [6]) is an intermediate modeling formalism with high-level language constructs suitable for representing complex engineering models. Nevertheless, it is compatible with model checking algorithms that are usually based on low-level formal models.

TXSTS models contain *data variables* to represent data-dependent behavior, while timed behavior is modeled by *clock variables*.

Clock variables are continuous, non-negative variables. They are initialized to zero and incremented equally. Most timed analysis techniques allow the comparison of clocks with other clocks or integer constants, and resetting them to integer constants. If there are rational constants in the model, all constants should be multiplied by the least common multiple of denominators [2]. For a set of clocks C, let $\mathcal{G}(C)$ denote the set of clock constraints in the form of $c_i \sim n$ or $c_i - c_j \sim n$, where $c_i, c_j \in C$, $\sim \in \{<, \leq, ==, \geq, >\}$, and $n \in \mathbb{N}_0$. Furthermore, let $\mathcal{A}(C)$ denote the set of clock assignments in the form of $c_i := n$, where $c_i \in C$, and $n \in \mathbb{N}_0$.

A timed extended symbolic transition system is a tuple $TXSTS = \langle V_D, V_C, V_{ctrl}, val^0, init, env, tran \rangle$ where

- V_D and V_C are finite sets of data variables and clock variables, $V_{ctrl} \subseteq V_D$ is a set of *control variables* that may be handled differently by the algorithms;
- val^0 is the initial valuation over V_D that maps each variable $x \in V_D$ to the initial value of the variable, or $\top$ if unknown;
- $init, env, tran \subseteq \mathcal{O}$ are sets of operations representing the *initialization*, *environment* and *internal operation sets*.

The operation sets consist of one or more operations taken from a set of operations $\mathcal{O}$. When executing an operation set, the operation to be executed is selected from the operation set in a nondeterministic manner. The set of operations $\mathcal{O}$ contains the following types of operations:

- *Assumptions* have the form $[\varphi]$, where φ is a Boolean combination of predicates over V_D and clock constraints of $\mathcal{G}(V_C)$;
- *Data assignments* have the form $x := \varphi$, where $x \in V_D$, and φ is an expression of the same type as x, that may contain variables of V_D and clock constraints of $\mathcal{G}(V_C)$;
- *Clock resets* are clock assignments of $\mathcal{A}(V_C)$;
- *Havoc operations* are denoted by $havoc(x)$, which is a nondeterministic assignment to a data variable $x \in V_D$;
- *Delays* are denoted simply by *delay*, it is a nondeterministic but equal incrementation of all clocks in V_C;
- A *no-op* is denoted by *skip*;
- A *sequence* is a composite operation $op_1, op_2, \ldots, op_n$, where $op_i \in \mathcal{O}$ for all $1 \leq i \leq n$, the operations are executed one after the other;
- *Nondeterministic choices* have the form $\{op_1\}$ *or* $\{op_2\}$ *or* ... *or* $\{op_n\}$, they are composite operations, where $op_i \in \mathcal{O}$ for all $1 \leq i \leq n$, from which exactly one operation is executed, chosen in a nondeterministic manner;
- *Conditional operations* of the form *if*(φ) *then* $\{op_1\}$ *else* $\{op_2\}$ are composite operations, where φ is a Boolean combination of predicates over V_D and clock constraints of $\mathcal{G}(V_C)$, if φ holds then $op_1 \in \mathcal{O}$ is executed, otherwise $op_2 \in \mathcal{O}$;
- A *loop* is a composite operation of the form *for* i *from* φ_a *to* φ_b *do* $\{op\}$, where i is an integer variable, φ_a, φ_b are expressions that evaluate to integers, serving as the lower and upper bound for the loop variable i, and $op \in \mathcal{O}$.

A *state* of a TXSTS model is a tuple $\langle val_D, val_C, \tau \rangle$, where val_D is a valuation over V_D, val_C is a valuation over V_C and $\tau \in \{init, env, tran\}$ is an operation set, which is the only operation set that can be executed in this state.

The order of execution of the operation sets is fixed in TXSTS models. The operation set *init* is executed only once, in the initial state. Sets *env* and *tran* are executed in an alternating manner, but only after *init*, starting with *env*.

The semantics of TXSTS operations is straightforward, however, one can refer to [4] for the detailed semantics.

When components of the same system are modeled as separate TXSTS models, we consider them as a *network of TXSTS models*, where time advances equally, as introduced in [10]. We will denote the state of a network $N = \langle TXSTS_1, TXSTS_2, \ldots, TXSTS_n \rangle$ by $S_N = \langle S_1, S_2, \ldots, S_n \rangle$, where S_i is a state of $TXSTS_i$ for each $1 \leq i \leq n$.

3. Formal modeling of coordination

In this section, we propose a new formalism, the *coordination automata*, to ease the challenges of modeling the possible execution orders of the system. We present the semantics of the coordination automata using the TXSTS modeling formalism, which is an extension of the inner modeling formalism used by the Gamma Framework. Finally, we present the coordination automaton of the motivating example.

3.1. Coordination automata

We propose the new coordination automata formalism, which extends timed automata [2] with notations referencing other formal models, e.g., TXSTS models. These models can be referenced on the edges of the coordination automaton, denoting that the given component is scheduled for execution.

A coordination automaton over the set of clocks C disjoint from the sets of clock variables of the referenced formal models and a set of TXSTS models $\mathcal{M} = \{TXSTS_1, TXSTS_2, \ldots, TXSTS_n\}$ is a tuple $CA = \langle L, l_0, E \rangle$, where

- L is a finite set of locations, with initial location $l_0 \in L$;
- $E \subseteq L \times \mathcal{M} \times 2^{\mathcal{G}(C)} \times 2^{\mathcal{A}(C)} \times L$ is a set of directed edges.

The semantics of coordination automata is similar to that of timed automata. With the set of TXSTS models $\mathcal{M}$ forming a network N, the semantics of the corresponding coordination automaton is defined by a transition system with a set of states $\mathcal{S}_{CA} \subseteq L \times Val_C \times \mathcal{S}_N$ where Val_C is the set of clock valuations over C, and $\mathcal{S}_N$ is the set of states of N.

The initial states of the transition system form a set $\{\langle l_0, val_C^{\Delta(S_N)}, S_N \rangle \mid S_N \in \mathcal{S}_N^{init}, \forall c \in C\colon val_C^{\Delta(S_N)}(c) = \Delta(S_N)\}$ where $\mathcal{S}_N^{init}$ is the set of possible states of N after executing the *init* operation set of all TXSTS models of N in the order given by the definition of the network, during which $\Delta(S_N)$ delay takes place if the resulting state of the network is S_N.

In the transition system there are two kinds of transitions:

- An action transition $\langle l, val_C, S_N \rangle \xrightarrow{m,G,A} \langle l', val'_C, S'_N \rangle$, where $m \in \mathcal{M}$, $G \subset \mathcal{G}(C)$ and $A \subset \mathcal{A}(C)$, is enabled iff the following conditions are satisfied:
  - $\langle l, m, G, A, l' \rangle \in E$;
  - val_C satisfies all clock constraints in G;
  - $\forall\ c \in C\colon\ val'_C(c) = z$ if $(c := z) \in A$, otherwise $val'_C(c) = val_C(c) + \Delta$, where $\Delta \in \mathbb{R}_{\geq 0}$ would be the value of a non-resettable clock in S'_N that was set to 0 in S_N;
  - S'_N is a result of executing the *env* and *tran* operation sets of m on S_N.
- A delay transition $\langle l, val_C, S_N \rangle \xrightarrow{\Delta \in \mathbb{R}_{\geq 0}} \langle l', val'_C, S'_N \rangle$ is enabled iff:

- $l' = l$;
- $\forall\ c \in C\colon\ val'_C(c) = val_C(c) + \Delta$; and
- $\forall\ m \in \mathcal{M},\ c \in V_C^m\colon\ val'_m(c) = val_m(c) + \Delta$, where V_C^m is the set of clock variables of m, and val_m, val'_m are the clock valuations of m in S_N and S'_N, respectively.

3.2. Sequential and unordered execution

To allow for more compact representation of the desired coordination of a distributed system, we provide two syntactic constructs for expressing the sequential and unordered execution of multiple components on a single edge of the coordination automaton.

The sequential execution, denoted as $seq\{m_1, m_2, \ldots, m_n\}$ on an edge from l_i to l_j, is equivalent to a path from l_i to l_j of length n, where the edges are annotated with components $m_1, m_2, \ldots, m_n$, respectively, with guards of the original sequential edge repeated on all edges along the path, and the last edge of the path containing the clock assignments of the original edge. I.e., the clock assignments are executed after the sequential execution of all referenced components, while the guards hold invariably until the scheduling of the last component.

Unordered execution is denoted as $unord\{m_1, m_2, \ldots, m_n\}$. Such an unordered execution on an edge from l_i to l_j is equivalent to having sequential paths from l_i to l_j for all permutations of $m_1, m_2, \ldots, m_n$. As with sequential paths, guards of the original unordered edge are repeated on all edges, and the last edges of the paths contain the clock assignments of the original edge. One should note that unordered execution results in $n!$ possible orderings of components, therefore advanced techniques are required for a systematic analysis of the system. Nonetheless, it provides a compact and easy-to-understand way for engineers to represent and communicate the coordination of a complex system.

4. Formal modeling of a distributed reactive system with coordinated components

The systematic examination of a system requires a formal representation. In this section, we describe a transformation of coordination automata and its referenced TXSTS models to a single TXSTS model, which results in a suitable input for model checking algorithms.

The idea of the transformation is mapping the coordination automaton to the environment operation set of an "integrated" TXSTS. The initialization (*init*), environment (*env*) and internal (*tran*) operation sets of the individual components referenced by the coordination automaton are then embedded in the corresponding operation sets of the integrated TXSTS. In a step of this TXSTS, only one of the components is selected for execution, based on a variable that is set by the transformed coordination automaton.

The mapping of the coordination automaton to the TXSTS formalism introduces at least two new control variables. The new variable *scheduled* represents the component that should be executed next by the *env* and *tran* operation sets. The second new variable, *coordState*, represents the current location of the coordination automaton, with initial value l_0. It may also stand for intermediate locations that are introduced by the transformation and represent the ongoing execution of a sequential or unordered edge.

4.1. Operation sets of the integrated TXSTS model

Let $init_m$, env_m and $tran_m$ denote the initialization, environment and internal operation sets of the TXSTS model $m \in \mathcal{M}$. We will also write $init_m$, env_m and $tran_m$ to denote a nondeterministic choice operation (see Subsection 2.2) of all operations in the given operation set. This will be used to represent an operation set as a single (nondeterministic) operation.

The *init* operation set of the integrated TXSTS model is a sequence of the *init* operation sets of all components: $init_{m_1}, init_{m_2}, \ldots, init_{m_{|\mathcal{M}|}}$.

The environment operation set is a sequence of two operations: a step of the coordination automaton and a step of one of the components.

The step of the coordination automaton is a sequence of a *delay* operation and a nondeterministic choice. In this nondeterministic choice, there is a branch for each location (both intermediate locations and locations of the coordination automaton). Each branch starts with the assumption $[coordState == l]$ (thus only one branch can be executed, determined by the value of *coordState*), which is followed by another nondeterministic choice. In this embedded nondeterministic choice, each branch represents an outgoing edge from location l.

I.e., if the outgoing edges of a location l_i are $e_{i,1}, e_{i,2}, \ldots, e_{i,n_i}$ and $(\!|e|\!)$ means the representation of an edge e in the TXSTS formalism, then the mapping of a step of the coordination automaton with locations l_i ($1 \leq i \leq k$, including intermediate locations) is the following:

$$\begin{aligned} &delay,\ \{\ [coordState == l_0],\ (\!|e_{0,1}|\!)\ or\ (\!|e_{0,2}|\!)\ or\ \ldots\ or\ (\!|e_{0,n_0}|\!)\ \} \\ &\quad or\ \{\ [coordState == l_1],\ (\!|e_{1,1}|\!)\ or\ (\!|e_{1,2}|\!)\ or\ \ldots\ or\ (\!|e_{1,n_1}|\!)\ \}\ or\ \ldots \\ &\quad\quad \ldots\ or\ \{\ [coordState == l_k],\ (\!|e_{k,1}|\!)\ or\ (\!|e_{k,2}|\!)\ or\ \ldots\ or\ (\!|e_{k,n_k}|\!)\ \}. \end{aligned}$$

The mapping of an edge e from l to l' with clock constraints $G \subset \mathcal{G}(C)$ and clock assignments $A \subset \mathcal{A}(C)$ depends on the kind of execution it represents.

If e is annotated by a *single* component $m \in \mathcal{M}$, then it is mapped to

$$[\bigwedge_{g \in G} g],\ A,\ scheduled := m,\ coordState := l',$$

i.e., the clock constraints of G become an assumption, the clock assignments of A are executed, m is marked for scheduling, and *coordState* is set to the target location of e.

If $e \in E$ is a *sequential or unordered edge*, then the domain of *coordState* is extended by a new intermediate location l_e, representing the ongoing execution of e. The location l_e is considered to have only one outgoing edge, to the target location of e, with the same guards, clock assignments and component notations (including *seq* and *unord* notations) as e. The intermediate locations ensure that the execution of sequential and unordered edges cannot be interrupted.

If e is a *sequential edge* annotated by $seq\{m_{s1}, m_{s2}, \ldots, m_{sn}\}$, then a new control variable seq_e is created, representing the number of executed components, and e is mapped to

$$[\underset{g\in G}{\wedge} g],\ coordState := l_e,\ seq_e := seq_e + 1,$$
$$if(seq_e == 1)\ then\ \{scheduled := m_{s1}\},$$
$$if(seq_e == 2)\ then\ \{scheduled := m_{s2}\},$$
$$\ldots,\ if(seq_e == n)\ then\ \{A,\ scheduled := m_{sn},\ coordState := l',\ seq_e := 0\}$$

where the *else* branches of conditional operations were omitted for simplification. The variable seq_e determines the component that should be executed next in the given sequence. When the last component is scheduled, the clock assignments are executed and *coordState* is set to the target location.

If e is an *unordered edge* annotated by $unord\{m_{u1}, m_{u2}, \ldots, m_{un}\}$, then a new control variable $unord_e$ is created (representing the component that should be scheduled next), as well as n Boolean control variables $unord_{e1}, unord_{e2}, \ldots,$ $unord_{en}$ (representing whether $m_{u1}, m_{u2}, \ldots, m_{un}$ has already been executed). The unordered edge e is then mapped to

$$[\underset{g\in G}{\wedge} g],\ coordState := l_e,\ havoc(unord_e),\ \{choose_1\}\ or\ \ldots\ or\ \{choose_n\},$$
$$if(\underset{1\le i\le n}{\wedge} unord_{ei})\ then$$
$$\{A,\ coordState := l',\ unord_{e1} := false,\ \ldots,\ unord_{en} := false\},$$

where $choose_i$ schedules the not yet executed m_{ui} for each $1 \le i \le n$:

$$[unord_e == m_{ui}\ \wedge\ \neg unord_{ei}],\ unord_{ei} := true,\ scheduled := m_{ui}.$$

Since some branch of the nondeterministic $\{choose_1\}$ *or* ... *or* $\{choose_n\}$ operation must be executed and all branches are constrained by assumptions of the form $\neg unord_{ei}$, the only feasible assignments at $havoc(unord_e)$ are those that represent a component that has not yet been executed. So, the component chosen for scheduling is certainly one of the eligible components. Lastly, if all components have already been scheduled, then the clock assignments are executed, and *coordState* is set to the target location.

The second part of the *env* operation set is the execution of the *env* set of one of the components, determined by *scheduled*. More precisely, it is a sequence

of a *delay* operation and conditional operations of the form $if(scheduled == m)\ then\ \{env_m\}$ for each $m \in \mathcal{M}$.

The internal transition set *tran* is constructed similarly. It starts with a *delay* operation, followed by conditional operations of the form $if(scheduled == m)\ then\ \{tran_m\}$ for each $m \in \mathcal{M}$. Therefore, the variable *scheduled* determines both the next environment step and the next internal step of the system.

4.2. Coordination of the SbW system

The coordination automaton of the motivating example is presented in Figure 2. The $q_0 \to q_3$ edge represents that, at first, the sensor components run and process their inputs. The sensor components must be executed in a one second long execution window, representing the possible deviation between their clocks. Since they are deployed separately and cannot influence each other's outputs, this behavior is modeled with the unordered execution. After all the sensors produced outputs, either the PrimaryRWA runs first and the SecondaryRWA runs second ($q_3 \to q_1$ and $q_1 \to q_0$ edges), or vice versa ($q_3 \to q_2$ and $q_2 \to q_0$ edges). This ordering can influence the cross-checking between the RWAs and possibly the MSP.

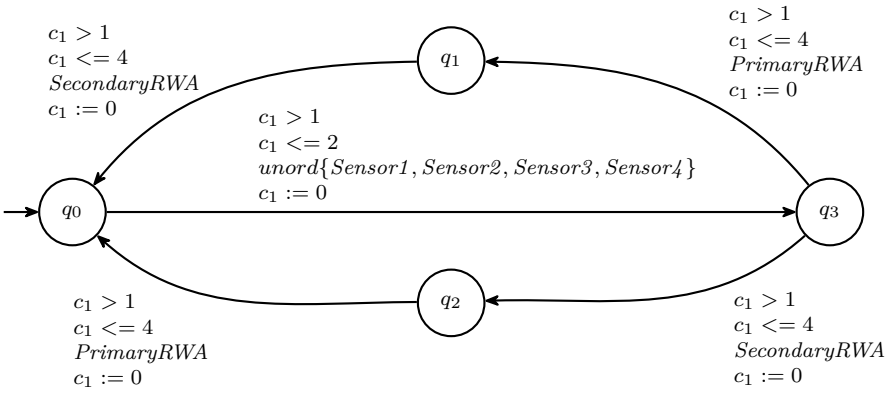


Figure 2. Coordination Automaton of the SbW System.

The TXSTS models representing the components of the system are connected in a single TXSTS model that already implements the coordination described by the given coordination automaton.

In locations q_1 and q_2 of the coordination automaton, there is only one possible execution order; in these cases, the mapping to TXSTS is straightforward, as shown in Listing 1 for location q_1.

In location q_3 of the coordination automaton there are multiple edges to other locations, referencing single TXSTS models. The mapping of this case to TXSTS is done using a nondeterministic choice for the possible outcomes, as shown in Listing 2.

```
assume coordState == q1;
assume c1 > 1 && c1 <= 4;
c1 := 0;
scheduled := SecondaryRWA;
coordState := q0;
```

Listing 1. TXSTS representation of deterministic scheduling of components in location q_1 of the coordination automaton

```
assume coordState == q3;
choice {
    assume c1 > 1 && c1 <= 4;
    c1 := 0;
    scheduled := PrimaryRWA;
    coordState := q1;
} or {
    assume c1 > 1 && c1 <= 4;
    c1 := 0;
    scheduled := SecondaryRWA;
    coordState := q2;
}
```

Listing 2. TXSTS representation of nondeterministically choosing the scheduled component in location q_3

```
assume coordState == q0; // q0q3 in case of line 18 in Listing 4.
assume c1 > 1 && c1 <= 2;
coordState := q0q3;
havoc unord_q0q3;
choice {
    assume unord_q0q3 == Sensor1 && !unord_q0q3_Sensor1;
    unord_q0q3_Sensor1 := true;
    scheduled := Sensor1;
} or {
    assume unord_q0q3 == Sensor2 && !unord_q0q3_Sensor2;
    unord_q0q3_Sensor2 := true;
    scheduled := Sensor2;
} or {
    ... // Sensor3
} or {
    ... // Sensor4
}
if (unord_q0q3_Sensor1 && unord_q0q3_Sensor2 && unord_q0q3_Sensor3 && unord_q0q3_Sensor4) {
    c1 := 0;
    coordState := q3;
    unord_q0q3_Sensor1 := false;
    unord_q0q3_Sensor2 := false;
    ... // Sensor3, Sensor4
}
```

Listing 3. TXSTS representation of unordered scheduling of sensors in location q_0 of the coordination automaton

For the unordered execution starting from location q_0 of the coordination automaton we introduce the variable `unord_q0q3` to represent the next scheduled component, which can take `Sensor1`, `Sensor2`, `Sensor3` or `Sensor4` as its value. We also introduce Boolean variables `unord_q0q3_Sensor1`, `unord_q0q3_Sensor2`, `unord_q0q3_Sensor3` and `unord_q0q3_Sensor4` that represent whether the given component was already scheduled. The scheduled component is chosen by a nondeterministic assignment to `unord_q0q3`, but considering only the components that were not yet scheduled, based on the newly introduced Boolean variables. The coordination automaton completes the transition to q_3 only when all referenced components were already scheduled. The TXSTS mapping of this unordered exe-

cution can be seen in Listing 3. The mapping of the intermediate location q_0q_3 is the same, except for the first line, where q_0 should change to q_0q_3.

The main structure of the resulting TXSTS model can be seen in Listing 4. The above-described mapping of the coordination automaton to the TXSTS formalism is embedded at the beginning of the single operation contained by the *env* operation set, followed by the operations of the individual components. If the *env* operation set of some component contains multiple operations (i.e., $env_{comp} = \{env_1, env_2, \ldots, env_n\}$), then these operations are wrapped in a nondeterministic choice $\{env_1\}$ *or* $\{env_2\}$ *or* ... *or* $\{env_n\}$, to represent them as a single operation without changing the possible behaviours of the model. The *tran* operation set is constructed analogously to the second part of the *env* operation set.

```
ctrl var coordState : enum{q0, q1, q2, q3, q0q3} = q0
ctrl var scheduled : enum{Sensor1, Sensor2, Sensor3,  Sensor4, PrimaryRWA, SecondaryRWA}
ctrl var unord_q0q3 : enum{Sensor1, Sensor2, Sensor3, Sensor4}
ctrl var unord_q0q3_Sensor1, unord_q0q3_Sensor2, unord_q0q3_Sensor3,
        unord_q0q3_Sensor4 : boolean = false
...
init {
    <init of Sensor1>
    <init of Sensor2>
    ... // Sensor3, Sensor4, PrimaryRWA, SecondaryRWA
}
env {
    delay;
    choice { assume coordState == q0; ... } // see Listing 3.
        or { assume coordState == q1; ... } // see Listing 1.
        or { assume coordState == q2; ... } // analogous with q1
        or { assume coordState == q3; ... } // see Listing 2.
        or { assume coordState == q0q3; ... } // see Listing 3.
    delay;
    if (scheduled == Sensor1) { <env of Sensor1> }
    if (scheduled == Sensor2) { <env of Sensor2> }
    ... // Sensor3, Sensor4, PrimaryRWA, SecondaryRWA
}
tran {
    delay;
    if (scheduled == Sensor1) { <tran of Sensor1> }
    if (scheduled == Sensor2) { <tran of Sensor2> }
    ... // Sensor3, Sensor4, PrimaryRWA, SecondaryRWA
}
```

Listing 4. Structure of a TXSTS model that schedules multiple TXSTS components based on a coordination automaton

5. Conclusion and future work

In this paper, we presented the coordination automata formalism as an extension of the timed automata formalism to model the possible interactions of the distributed subsystems. The presented coordination automata formalism is flexible and configurable: the coordinated subsystems can be modeled using arbitrary formal or engineering modeling languages.

As a continuation of our work, we investigate the possibilities of extending the coordination automata with more complex parallelism, to define coordination with overlapping component executions.

References

[1] K. Altisen, G. Gössler, J. Sifakis: *Scheduler modeling based on the controller synthesis paradigm*, Real-Time Systems 23 (2002), pp. 55–84, doi: 10.1023/A:1015346419267.

[2] R. Alur: *Timed automata*, in: Computer Aided Verification: 11th International Conference, CAV'99 Trento, Italy, July 6–10, 1999 Proceedings 11, Springer, 1999, pp. 8–22, doi: 10.1007/3-540-48683-6_3.

[3] C. Baier, J. Katoen: *Principles of model checking*, MIT Press, 2008.

[4] D. Cziborová: *Abstraction-based model checking for real-time software-intensive system models*, Scientific Students' Association Report, Budapest University of Technology and Economics, 2023.

[5] D. Cziborová, R. Szabó: *Modeling of Time-Dependent Behavior in Fault-Tolerant Systems*, in: 31th PhD Minisymposium of the Department of Measurement and Information Systems, Budapest University of Technology and Economics, 2024, doi: 10.3311/MINISY2024-011.

[6] B. Graics, M. Mondok, V. Molnár, I. Majzik: *Model-Based Testing of Asynchronously Communicating Distributed Controllers*, in: Formal Aspects of Component Software, ed. by J. Cámara, S.-S. Jongmans, 2024, pp. 23–44, isbn: 978-3-031-52183-6, doi: 10.1007/978-3-031-52183-6_2.

[7] M. Lohstroh, C. Menard, S. Bateni, E. A. Lee: *Toward a lingua franca for deterministic concurrent systems*, ACM Tran. Embedded Comput. Syst. 20.4 (2021), pp. 1–27, doi: 10.1145/3448128.

[8] V. Molnár, B. Graics, A. Vörös, I. Majzik, D. Varró: *The Gamma Statechart Composition Framework: design, verification and code generation for component-based reactive systems*, in: ICSE '18, ACM, 2018, pp. 113–116, doi: 10.1145/3183440.3183489.

[9] C. Norstrom, A. Wall, W. Yi: *Timed automata as task models for event-driven systems*, in: Proceedings Sixth International Conf. on Real-Time Computing Systems and Applications. RTCSA'99 (Cat. No. PR00306), IEEE, 1999, pp. 182–189, doi: 10.1109/RTCSA.1999.811218.

[10] R. Szabó, D. Cziborová, A. Vörös: *Towards Configurable Coordination for Distributed Reactive Systems*, in: 32th PhD Minisymposium of the Dept. of AI and Systems Engineering, Budapest University of Technology and Economics, 2025, doi: 10.3311/MINISY2025-009.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 214–225
DOI: 10.17048/fmfai.2025.214

Perimeter defense game with nonzero capture radius in a circular target

Sára Szénási*, **István Harmati**

Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics, Budapest
{szenasi,harmati}@iit.bme.hu

Abstract. In this paper, the problem of guarding a circular target wherein the Defender is constrained to move along its perimeter and has nonzero capture radius is posed and solved using a differential game theoretic approach. The Perimeter Defense Game is a special case of Pursuit-Evasion Game, where the goal of the pursuer is capturing the evader. In the Perimeter Defense Game the Attacker seeks to reach the perimeter of the circular target, whereas the Defender seek to align itself with the Attacker, thereby ending the game. The Defender has nonzero capture radius, which means that the Defender wins, when the distance between the Attacker and the Defender is smaller than the value of the capture radius. The Perimeter Defense Game can be divided into two cases: Win of Defender and the Win of Attacker scenarios. In the case when the Defender wins, the agents play a zero sum differential game, where the cost/payoff is the Attacker's terminal distance to the target. In the case when the Attacker wins, the agents play a zero-sum differential game, where the payoff/cost is the distance between the Defender and the Attacker. The analytic solutions of optimal strategies and the winning regions are also presented.

Keywords: differential game theory, Perimeter Defense Game, nonzero capture radius

AMS Subject Classification: 49N70 Differential games and control

*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, under a grant agreement with the National Research, Development and Innovation Office.

1. Introduction

In the reach avoid games there are two competitive teams, where one team attempts to arrive at a goal set in the state-space while avoiding some other undesired set of states. The goal of the opposing team is to prevent the first player from arriving at its goal [16]. The team consists of one or more players. The perimeter defense games are the special case of reach-avoid games. In a perimeter defense game the defender's team constrained to move along the convex perimeter and the attacker's team move with simple motion [10, 11]. The turret defense games are similar of perimeter defense games. In turret defense games the turret can be shoot the attacker from a certain distance. It can be transformated the game when the defender moves along a circular target and this way the problem can be solved easily using analytical methods [15]. The main difference between the perimeter defense and the transformated turret defense game is the winning condition of the attacker. During perimeter defense games, if the attacker reach the target at the same time as the defender makes an interception the defender wins, while in turret defense game the attacker emerge victorious.

The problem of guarding a target has many important application in real world. One example is protection of a building's perimeter against a sequentially arriving intruder [7]. In a real world there are not always information about the full state space, therefore, the information must be collected beforehand, for example with patrolling agents [12]. The target guarding can be applied in three dimension also, for example in the perimeter-defense game between aerial defender and ground intruder [6]. The survival is also a possible application, where one agent want to reach a safety zone while one [14] or more turret [4] want to neutralize its. Turret defense game with non-zero neutralization angle is also solved [8].

The perimeter defense game has many variant depending on the number of defenders, the number of intruders and the goal of the intruder(s). In a basic scenario there is one intruder and one or two defenders and the goal of the intruder is maximalizing the angular separation from the defender(s) when its reach the target [15]. The direct generalization is the perimeter defense game with more defenders and more intruders. This game can be solved with splitting into subgames with one or two defenders and one attacker [9, 10, 13].

In the reach avoid game the capture radius plays important role. The geometric solution of a target defense game where the attacker and also a defender can move freely in a full state space region with faster defenders with non-zero capture radius and a convex target area using the Hamilton-Jacobi-Isaacs equation is proven, but only partially appliable to our research, because of the constrained movements of the defender [3]. But in most of the existing paper discussing the perimeter defense games, the defender can capture with zero capture radius. There are proposals mentioned for the solution for the case of non-zero capture radius [12], but it is not complete proven and using only geometric solution. The most notable difference in this paper compared to the previous results that we studied the case when the capture radius is non-zero. In this analysis, we use the first order of necessary

conditions for optimality according to a classical differential game approach [5]. The steps of solutions is identical to the basic turret defense game [15] with different termination and optimalization constraints.

The main contribution of this paper is the analytic solution with use of differential game approach of the Perimeter Defense Game with nonzero capture radius in a circular target. We give a step by step exploration of the solution.

The paper constructs as follows. After this introduction in Section 1, Section 2 presents the problem statement. Followed by Section 3, where the steps of the solution method are demonstrated. In section 4 the results are shown and the equilibrium flow field is presented. Last, in section 5 the conclusion and the future works with possible research directions are explored.

2. Problem statement

This paper formulates the target guarding problem wherein the Defender (D) constrained to move along the circular target perimeter and the Attacker (A) moves in the plane with simple motion. The Defender can make interception with r capture radius. In figure 1 can be seen the illustration and the rules of the game: R denotes the distance between the target center and the attacker, θ denotes the angular separation between the defender and the attacker and β the angular of the defender referred to the x axis. The goal of the Attacker is to enter the target, reach $\mathcal{T}$ without interception, and the goal of the Defender is preventing breach by the Attacker. Selected assumptions are made on the problem statement:

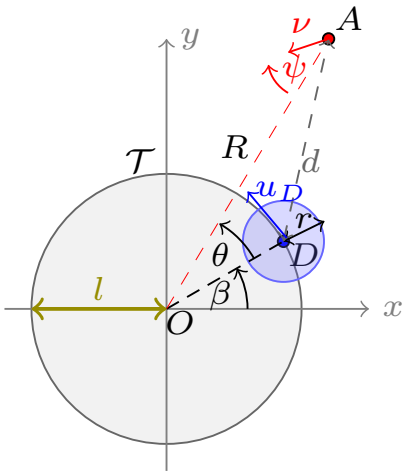


Figure 1. The illustration of the perimeter defense game with nonzero capture radius.

Assumption 2.1. *The target is a circle with $l = 1$ radius.*

Assumption 2.2. *The player's speeds are such that $0 < \nu \leq u_D = 1$, where ν is the speed of Attacker and u_D is the speed of Defender.*

Assumption 2.3. *The Defender makes interception with r capture radius. The $\mathcal{C}$ Capture Circle is defined as the set of the states of satisfying*

$$\mathcal{C} = \{(R, \theta) \,|\, r^2 \geq R^2 + 1 - 2R\cos\theta\} \tag{2.1}$$

Assumption 2.4. *The initial separation angle is such that $\theta(t_0) = \theta_0 \in [0,\, \pi)$*

Remark 2.5. The solution in case when $\theta_0 \in [-\pi, 0)$ can be determined from the symmetry result.

Assumption 2.6. *The initial Attacker distance is such that $R(t_0) > 1$, that is, A begins outside the target circle.*

Assumption 2.7. *The initial states are outside the Capture Circle*

$$(R_0, \theta_0) \notin \mathcal{C}. \tag{2.2}$$

The kinematics can be written as

$$f(\mathbf{x}, u, t) = \dot{\mathbf{x}} = \begin{bmatrix} \dot{R} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\nu\cos\psi \\ \nu\frac{1}{R}\sin\psi - u_D \\ u_D \end{bmatrix} \tag{2.3}$$

The Defender control is the value of the speed of the defender and lies in the range $u_D \in [-1,\, 1]$ and the defender speed direction is always the tangent of the $\mathcal{T}$ target perimeter. The Attacker control is the heading angle referred to the line between the target center and the attacker and lies in the range $\psi \in [-\pi\,, \pi]$.

2.1. Defender wins scenario

In the Win of Defender (WoD) scenario, when D is able to make interception before A can reach the target, the agents play zero sum game over the cost-functional

$$J_d := \Phi_d(\mathbf{x}_f, t_f) = -R_f, \tag{2.4}$$

where the subscript f denotes the termination, Φ_d denotes the terminal value function that depends on the decision of the attacker and the defender. So the cost functional is the negative Attacker's distance from the target center at the end of the game. The Defender is the minimizing player and the Attacker is the maximizing player. The Value of the game if it exists, is the saddle-point equilibrium of the cost-functional over state-feedback strategies

$$V_d = \min_{u_D(\cdot)} \max_{\psi(\cdot)} J_d = \max_{\psi(\cdot)} \min_{u_D(\cdot)} J_d. \tag{2.5}$$

The terminal constraint is

$$\phi_d(\mathbf{x}_f, t_f) = \sqrt{R^2 + 1 - 2R\cos\theta} - r = d - r = 0, \tag{2.6}$$

that means that the game is terminated, if the distance between the Defender and the Attacker equals to the r capture radius. The final time t_f is the first time for which $d = r$. Thus, the Terminal Surface is defined as the set of states of satisfying (2.4)

$$\mathcal{J}_d = \{\mathbf{x} \,|\, R > 1 \quad \text{and} \quad d = r\}. \tag{2.7}$$

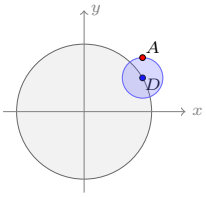


Figure 2. The illustration of the Defender wins scenario radius.

2.2. Attacker wins scenario

In Win of Attacker (WoA) scenario, when A is able to drive $R = 1$ while avoiding $d \leq r$, because of the separation angle is proportional the distance of the agents in case $R_f = 1$, the agents play zero sum game over the cost-functional:

$$J_a := \Phi_a(\mathbf{x}_f, t_f) = \theta_f - \theta_r, \tag{2.8}$$

where θ_r is the angular separation in the limiting case if $d_f = r$ and $R_f = 1$ it can be determined from theorem of cosines from AOD triangle in figure 1 $\theta_r = \arccos\left(\frac{2-r^2}{2}\right)$. The Defender is the minimizing player and the Attacker is the maximizing player. The Value of the game if it exists, is the saddle-point equilibrium of the cost-functional over state-feedback strategies

$$V_a = \min_{u_D(\cdot)} \max_{\psi(\cdot)} J_a = \max_{\psi(\cdot)} \min_{u_D(\cdot)} J_a. \tag{2.9}$$

Termination occurs when the Attacker reaches the target circle, therefore the termination constraint

$$\phi(\mathbf{x}_f, t_f) = R_f - 1 = 0. \tag{2.10}$$

The final time t_f is the first time for which $R(t) = 1$. Thus, the Terminal Surface is defined as the set of states of satisfying (2.10)

$$\mathcal{J}_a = \{\mathbf{x} \,|\, R = 1 \quad \text{and} \quad d \geq r\}. \tag{2.11}$$

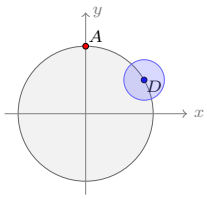


Figure 3. The illustration of the Attacker wins scenario.

3. Methods

The steps of the analytic solution of the Defender wins scenario (WoD) follow the steps of the Turret Defense Game [15], but we form the steps to our problem statement. We use the cost function (2.4) and the terminal constraint (2.6) during the derivation. The analysis is carried out according to a classical differential game approach [1, 5]. The solution of Attacker wins scenario (WoA) based upon showing satisfaction of the sufficient condition. The proposed equilibrium strategies of the Defender wins scenario and Value function substituting into the Hamilton-Jacobi-Isaacs equation [5].

3.1. Solution of Defender wins scenario

The steps of the following analytic solution follow the steps of the solution of zero-capture radius case [15] with subtituting the cost function (2.4) and terminal constraint (2.6).

The analysis is carried out according to a classical differential game approach [1, 5]. The Hamiltonian of the Defender wins scenario is

$$\mathcal{H}_d = -\sigma_R \nu \cos\psi + \sigma_\theta \left(\nu \frac{1}{R} \sin\psi - u_D \right) + \sigma_\beta u_D \tag{3.1}$$

where $\sigma \equiv [\sigma_R \quad \sigma_\theta \quad \sigma_\beta]^T$ is the adjoint vector. The Hamiltonian is a separable function of the controls u_D and ψ, and thus Isaacs' s condition [5] holds:

$$\min_{u_D(\cdot)} \max_{\psi(\cdot)} \mathcal{H}_d = \max_{\psi(\cdot)} \min_{u_D(\cdot)} \mathcal{H}_d \quad \forall \mathbf{x} \tag{3.2}$$

The Defender minimalize and the Attacker maximalize the Hamiltonian. The Defender control range is $u_D \in [-1, \quad 1]$ and the Attacker control range is $\psi \in [-\pi, \quad \pi]$. The equilibrium adjoint dynamics are given by

$$\dot{\sigma}_R = -\frac{\partial \mathcal{H}_d}{\partial R} = \sigma_\theta \nu \frac{1}{R^2} \sin\psi \tag{3.3}$$

$$\dot{\sigma}_\theta = -\frac{\partial \mathcal{H}_d}{\partial \theta} = 0 \tag{3.4}$$

$$\dot{\sigma}_\beta = -\frac{\partial \mathcal{H}_d}{\partial \beta} = 0 \tag{3.5}$$

The terminal adjoint values are obtained from the transversality condition [2]

$$\sigma(t_f) = \frac{\partial \Phi_d}{\partial \mathbf{x}_f} + \eta \frac{\partial \phi_d}{\partial \mathbf{x}_f} \tag{3.6}$$

$$\begin{aligned} \sigma_{R_f} &= -1 + \tfrac{\eta}{r}(R_f - \cos\theta_f) \\ \Rightarrow \sigma_{\theta_f} &= \tfrac{\eta}{r} R_f \sin\theta_f \\ \sigma_{\beta_f} &= 0 \end{aligned} \tag{3.7}$$

where η is an additional adjoint variable. Therefore, with (3.3), (3.7), the following hold

$$\sigma_\theta(t) = \frac{\eta}{r} R_f \sin\theta_f \quad \forall t \in [t_0\,,\ t_f] \tag{3.8}$$

$$\sigma_\beta(t) = 0 \quad \forall t \in [t_0\,,\ t_f] \tag{3.9}$$

Since $\sigma_\beta(t) = 0$ for all $t \in [t_0\,,\ t_f]$, the state component β has no effect on the equilibrium trajectory or the equilibrium control strategies. The terminal Hamiltonian satisfies [2]

$$\mathcal{H}_d(t_f) = -\frac{\partial \Phi_d}{\partial t_f} - \eta \frac{\partial \phi_d}{\partial t_f} = 0 \tag{3.10}$$

Since Φ_d and ϕ_d independent on time and $\frac{d\mathcal{H}_d}{dt} = 0$ so $\mathcal{H}_d(t) = 0$, $t \in [t_0, t_f]$.

The equilibrium control actions of the Attacker and Defender maximize and minimize (3.2), respectively:

$$\mathcal{H}_d^* = \max_{\psi} \min_{u_D} \mathcal{H}_d. \tag{3.11}$$

In order to maximize (3.2), the vector $[\cos\psi \quad \sin\psi]$ must be parallel to the vector $[-\sigma_R \quad \frac{\sigma_\theta}{R}]$. Therefore the optimal control of the Attacker can be expressed as:

$$\cos\psi^* = \frac{-\sigma_R}{\sqrt{\sigma_R^2 + \left(\frac{\sigma_\theta}{R}\right)^2}} \qquad \sin\psi^* = \frac{\frac{\sigma_\theta}{R}}{\sqrt{\sigma_R^2 + \left(\frac{\sigma_\theta}{R}\right)^2}}. \tag{3.12}$$

If $\sigma_\theta < 0$, this implies $\sin\psi^* < 0$ due to (3.12). However, this would mean the Attacker has a component of its motion that points towards the Defender due to Assumption 2.4. Thus, it must be the case that $\sigma_\theta > 0$ In order to minimize (3.1), the Defender's control must satisfy

$$u_D^* = \operatorname{sign}\sigma_\theta = 1, \tag{3.13}$$

since $\sigma_\theta > 0$.

To express the adjoint variable η must be substituting the equilibrium controls, (3.12) and (3.13), into the Hamiltonian (3.1) and evaluating at final time with (3.7) and (3.10) gives

$$\mathcal{H}_d^*(t_f) = -\sigma_{R_f}\nu\cos\psi^* + \sigma_\theta\left(\frac{\nu}{R_f}\sin\psi^* - u_D^*\right) \tag{3.14}$$

$$\longrightarrow \nu\sqrt{\sigma_{R_f}^2 + \left(\frac{\sigma_\theta}{R_f}\right)^2} - \sigma_\theta = 0 \tag{3.15}$$

An expression for σ_R is obtained by considering the Hamiltonian at a general time and substituting the equilibrium controls (3.12) (3.13) into the Hamiltonian (3.1):

$$\mathcal{H}_d^*(t) = 0 = -\sigma_R\nu\cos\psi^* + \sigma_\theta\left(\nu\frac{1}{R}\sin\psi^* - u_D^*\right) \tag{3.16}$$

$$\longrightarrow \nu\sqrt{\sigma_R^2 + \left(\frac{\sigma_\theta}{R}\right)^2} - \sigma_\theta = 0 \tag{3.17}$$

$$\longrightarrow \sigma_R = \sqrt{\frac{\sigma_\theta^2}{\nu^2}\left(1 - \frac{\nu^2}{R^2}\right)} \tag{3.18}$$

The ψ^* optimal heading angle can be determined by substituting the adjoint variables to the equilibrium the attackers controls (3.12)

$$\cos\psi^* = \frac{-\sigma_R}{\sqrt{\sigma_R^2 + \left(\frac{\sigma_\theta}{R}\right)^2}} = \frac{-\frac{\sigma_\theta}{\nu}\sqrt{1 - \frac{\nu^2}{R^2}}}{\sqrt{\frac{\sigma_\theta^2}{\nu^2}\left(1 - \frac{\nu^2}{R^2}\right) + \frac{\sigma_\theta^2}{R^2}}} = \sqrt{1 - \frac{\nu^2}{R^2}} \tag{3.19}$$

$$\sin\psi^* = \frac{\frac{\sigma_\theta}{R}}{\sqrt{\sigma_R^2 + \left(\frac{\sigma_\theta}{R}\right)^2}} = \frac{\nu}{R} \tag{3.20}$$

The equilibrium kinematics can be obtained by substituting the equilibrium controls (3.19) and (3.13) into (2.3) which yields

$$\dot{R}^* = -\nu\cos\psi^* = -\nu\sqrt{1 - \frac{\nu^2}{R^2}} \tag{3.21}$$

$$\dot{\theta}^* = \nu\frac{1}{R}\sin\psi^* - u_D^* = \frac{\nu^2}{R^2} - 1 \tag{3.22}$$

with the following boundary conditions $R_f > 1$, $r^2 = R_f^2 + 1 - 2R_f\cos\theta_f$.

Considering the differential equation obtained by dividing the equations in (3.21)

$$\frac{dR}{d\theta} = \frac{\nu}{\sqrt{1 - \frac{\nu^2}{R^2}}} \tag{3.23}$$

$$\Rightarrow \nu \left[\sqrt{\frac{R^2}{\nu^2} - 1} + \arcsin\left(\frac{\nu}{R}\right) \right]_{R_f}^{R} = \nu(\theta - \theta_f) \tag{3.24}$$

Define

$$g(R) = \sqrt{\frac{R^2}{\nu^2} - 1} + \arcsin\left(\frac{\nu}{R}\right) \tag{3.25}$$

$$\Rightarrow \nu(g(R) - g(R_f)) = \nu(\theta - \theta_f) \tag{3.26}$$

$$\Rightarrow \theta(R; R_f, \theta_f) = g(R) - g(R_f) + \theta_f, \quad r^2 = R_f^2 + 1 - 2R_f \cos\theta_f \tag{3.27}$$

Setting different θ_f, $0 \leq \theta_f \leq \theta_r = \arccos\left(\frac{2-r^2}{2}\right)$ in (3.26) describes equilibrium flow field for the Defender wins scenario. The equilibrium flow field gives the equilibrium trajectory from given terminal states. The optimal attacker path is the involute of a circle with radius ν.

The symmetric solution if $\theta < 0, t \in [0, t_f]$ can be solved in a same way. If $\theta_0 = \pi$ called dispersal surface, and in this case the positive and negative solution results the same value of the game. If $\theta_0 = 0$ called afferent surface and in this case the defender optimal trajectory is keep the zero angular separation.

The equilibrium state feedback control strategies for the Defender wins scenario are given by

$$\psi^* = \text{sign}(\theta) \arcsin\left(\frac{\nu}{R}\right) \quad u_D^* = \text{sign}(\theta) \tag{3.28}$$

The expression for ψ^* is obtained by (3.19) taking into account the sign of θ. Similarly, the Defender strategy is given by (3.13) taking into the sign of θ.

The Value of the game is

$$V_d(R, \theta) = -R_f \tag{3.29}$$

$$g(R_f) = g(R) + \theta_f - |\theta| \tag{3.30}$$

$$\Rightarrow V_d(R, \theta) = -g^{-1}(g(R) + \theta_f - |\theta|) \tag{3.31}$$

where g is defined in (3.25). Because V_d is defined using the inverse of the function g, it is necessary to show that $g(R)$ is monotic. Taking the derivate of (3.25) w.r.t. R gives

$$\frac{dg}{dR} = \frac{\sqrt{R^2 - \nu^2}}{\nu R}, \tag{3.32}$$

It must be that $0 < \nu < 1$ from Assumption 2.2 and from Assumption 2.6 it must be that $R > 1$ throughout the game. So we have $R > \nu$ and $R, \nu > 0$ which implies that $g(R)$ is monotonic.

The Value function does not have a closed form analytic expression since g^{-1} cannot be expressed in closed form.

The limiting case for the Defender wins scenario is one in which $R_f \longrightarrow 1$, $\theta_f = \theta_r$; thus the surface

$$\theta_{GoK}(R) = g(R) - g(1) + \theta_r \tag{3.33}$$

partitions the state space into regions of win for the Defender and Attacker, respectively,

$$\mathcal{R}_D = \{\mathbf{x} | \, |\theta| \leq \theta_{GoK}(R)\} \tag{3.34}$$

$$\mathcal{R}_A = \{\mathbf{x} | \, |\theta| > \theta_{GoK}(R)\}. \tag{3.35}$$

3.2. Solution of Attacker wins scenario

The solution of Attacker wins scenario based upon showing satisfaction of the sufficient condition for equilibrium via substitution of the proposed equilibrium strategies and Value function into the Hamilton-Jacobi-Isaacs equation [5]. The equilibrium state feedback strategies for the Attacker wins scenario is match those of the Defender wins scenario. The Value function is given by

$$V(R, \theta) = \theta_f - \theta_r = \theta - g(R) + g(1) - \theta_r \tag{3.36}$$

The Hamilton-Jacobi-Isaacs equation can be written as [5]

$$\min_{u_D} \max_{\psi} \left\{ l(\mathbf{x}, u_D, \psi, t) + \frac{\partial V}{\partial t} + V_x f(\mathbf{x}, u_D, \psi, t) \right\} = 0 \tag{3.37}$$

where V_x is the vector $\left[\frac{\partial V}{\partial R} \quad \frac{\partial V}{\partial \theta} \quad \frac{\partial V}{\partial \beta}\right]^T$ and l represents an integral cost component. First, note that the cost, has no integral component, and thus $l = 0$. Also, the proposed Value function (3.36) is not an explicit function of time and thus $\frac{\partial V}{\partial t} = 0$. The vector V_x is obtained by differentiating (3.36) w. r. t. each state

$$V_x = \left[\frac{-\sqrt{R^2 - \nu^2}}{R\nu} \quad 1 \quad 0\right]. \tag{3.38}$$

The equilibrium dynamics f are given by (3.21). Substituting (3.38) , $\frac{\partial V}{\partial t} = 0$ and $l = 0$ into (3.37) gives

$$\frac{\partial V}{\partial R}\dot{R} + \frac{\partial V}{\partial \theta}\dot{\theta} = 0. \tag{3.39}$$

So the given Value function satisfies the Hamilton-Jacobi-Isaacs equation and this reason the equilibrium state feedback strategies are same of the cases Attacker and the Defender wins scenario The trajectories 3.26 are also same at the two cases, but in the Attacker wins scenario $r^2 < R_f^2 + 1 - 2R_f \cos\theta_f$ holds.

4. Results

In a Defender wins and Attacker wins scenario the agents have the same equilibrium strategies: the Attacker moves the tangent of the ν radius circle and the defender moves along the perimeter of the target towards the Attacker. The equilibrium flow field shows the trajectories in the (R, θ) plane for the Defender and Attacker wins scenario, also gives the terminal states and this way the winning regions. Figure 4

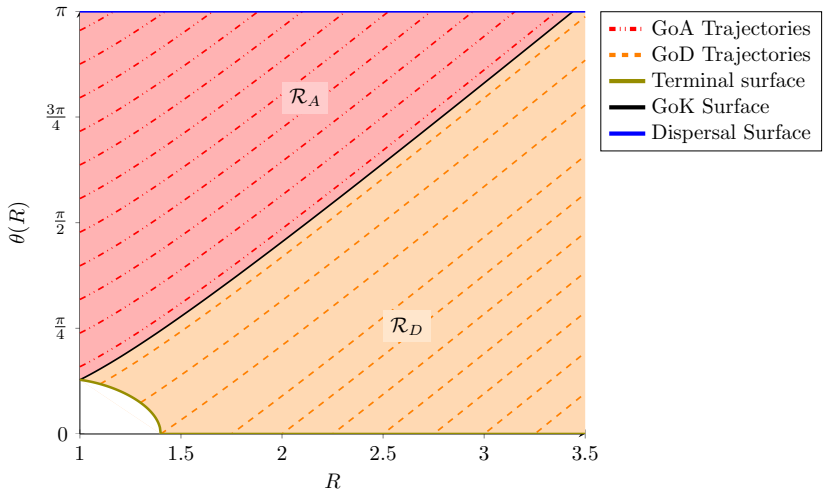


Figure 4. Full equilibrium flow field with $\nu = 0.8$ and $r = 0.4$.

shows the full equilibrium flow field in case $\nu = 0.8$, $r = 0.4$. The Attacker winning region and the trajectories denoted by red, the Defender winning region and the trajectories denoted by orange, the trajectory of limiting case denoted by black and the terminal surface of Defender wins scenario denoted by olive. The white region represents the $\mathcal{C}$ capture circle.

5. Conclusion and future works

In this paper we presented and solved the perimeter defense game with one Attacker and one Defender with r capture radius in a circular target. We created some assumptions and then showed the Defender and the Attacker win scenario and its solutions applying the Hamiltonian and the Hamilton-Jacobi-Isaacs equation ie. the first order necessary conditions for optimality and the sufficient condition for the equilibrium. The equilibrium state feedback strategies, the winning regions and the full equilibrium flow field are also presented. The equilibrium state feedback strategies are the same as the case of the point capture, but the winning regions and the equilibrium flow field depend on the r capture radius.

In future work we aim to solve the perimeter defense game with r capture radius applying more attackers and defenders. Perimeter defense game with general convex shape target, or general convex shape capture region is also a possible future work. It is also a possible generalization if $\nu > 1$, so the defenders have larger speed as the attackers, but there are more defenders than attackers. It is also possible future work when the Attacker(s) have penetration radius and this way they can reach the target earlier.

References

[1] T. Başar, G. J. Olsder: *Dynamic noncooperative game theory*, SIAM, 1998.

[2] A. E. Bryson: *Applied optimal control: Optimization*, Estimization and Control 2 (1975).

[3] H. Fu, H. H.-T. Liu: *Justification of the geometric solution of a target defense game with faster defenders and a convex target area using the HJI equation*, Automatica 149 (2023), p. 110811.

[4] Z. E. Fuchs, A. Von Moll, D. Casbeer: *Engage or retreat differential game with n-targets and distributed defensive assets*, in: 2021 IEEE Conference on Control Technology and Applications (CCTA), IEEE, 2021, pp. 386–393.

[5] R. Isaacs: *Differential Games, a Mathematical Theory with Applications to Optimization*, 1965.

[6] E. S. Lee, D. Shishika, V. Kumar: *Perimeter-defense game between aerial defender and ground intruder*, in: 2020 59th IEEE conference on decision and control (CDC), IEEE, 2020, pp. 1530–1536.

[7] A. Pourghorban, M. Dorothy, D. Shishika, A. Von Moll, D. Maity: *Target defense against sequentially arriving attackers*, in: 61st IEEE Conference on Decision and Control (forthcoming), 2022.

[8] I. H. Sára Szénási: *Turret Defense Game with Nonzero Neutralization Angle*, Proceedings of the Workshop on the Advances of Information Technology 2025 (2025), pp. 83–89.

[9] D. Shishika, V. Kumar: *A review of multi agent perimeter defense games*, in: International conference on decision and game theory for security, Springer, 2020, pp. 472–485.

[10] D. Shishika, V. Kumar: *Local-game decomposition for multiplayer perimeter-defense problem*, in: 2018 IEEE conference on decision and control (CDC), IEEE, 2018, pp. 2093–2100.

[11] D. Shishika, V. Kumar: *Perimeter-defense game on arbitrary convex shapes*, arXiv preprint arXiv:1909.03989 (2019).

[12] D. Shishika, J. Paulos, M. R. Dorothy, M. A. Hsieh, V. Kumar: *Team composition for perimeter defense with patrollers and defenders*, in: 2019 IEEE 58th Conference on Decision and Control (CDC), IEEE, 2019, pp. 7325–7332.

[13] D. Shishika, J. Paulos, V. Kumar: *Cooperative team strategies for multi-player perimeter-defense games*, IEEE Robotics and Automation Letters 5.2 (2020), pp. 2738–2745.

[14] A. Von Moll, Z. E. Fuchs: *Optimal constrained retreat within the turret defense differential game*, in: 2020 IEEE conference on control Technology and applications (CCTA), IEEE, 2020, pp. 611–618.

[15] A. Von Moll, M. Pachter, D. Shishika, Z. Fuchs: *Circular target defense differential games*, IEEE Transactions on Automatic Control 68.7 (2022), pp. 4065–4078.

[16] Z. Zhou, R. Takei, H. Huang, C. J. Tomlin: *A general, open-loop formulation for reach-avoid games*, in: 2012 IEEE 51st IEEE conference on decision and control (CDC), IEEE, 2012, pp. 6501–6506.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 226–232
DOI: 10.17048/fmfai.2025.226

Full-parameter fine-tuning vs. LoRA fine-tuning on PULI models

Kristóf Varga[a], **Péter Hatvani**[ab], **Zijian Győző Yang**[a]

[a]ELTE Research Centre for Linguistics
{varga.kristof,hatvani.peter,yang.zijian.gyozo}@nytud.elte.hu

[b]Pázmány Péter Catholic University
Doctoral School of Linguistics
hatvani.peter@hallgato.ppke.hu

Abstract. In this study, we compare full-parameter fine-tuning and parameter-efficient LoRA on various Hungarian PULI large language models, evaluating their performance across six Hungarian language understanding benchmarks. While full-parameter fine-tuning updates all model weights and requires substantial computational resources, LoRA adapts a smaller subset of parameters, enabling more efficient training. Our experiments on the monolingual PULI 3SX and the multilingual LlumiX and LlumiX-Llama-3.1 models reveal that LoRA consistently matches or surpasses full fine-tuning on most tasks, particularly when applied to larger models. Notably, LlumiX-Llama-3.1 with LoRA achieves state-of-the-art results on five out of six benchmarks while significantly reducing resource demands. These findings highlight LoRA's potential as a scalable and effective fine-tuning method for Hungarian large language models.

Keywords: LoRA, PULI models, HuLU benchmarks, fine-tuning, parameter-efficient adaptation

AMS Subject Classification: 68T07, 68T50, 68U15

1. Background

In recent months, large language models have undergone significant development and have become one of the most popular topics in the field of artificial intelligence. The training of language models consists of two well-defined phases: pretraining and fine-tuning. During pretraining, a neural network (most commonly based on the transformer architecture [8]) is trained on general language understanding. Af-

ter this phase, the model is further trained for task-specific knowledge through fine-tuning. In the case of large language models, fine-tuning can be used to train the model for conversational purposes or domain-specific knowledge. Although fine-tuning requires fewer resources than pre-training, it still requires significant hardware power when working with large language models. In traditional fine-tuning, all the parameters of the pretrained model are updated – this is called full-parameter fine-tuning. However, this approach is extremely resource intensive. To address this, parameter-efficient methods have been developed [3, 6, 11], among which one of the most popular is LoRA. LoRA (Low-Rank Adaptation) improves training efficiency in multiple ways; one key aspect is that it adapts a pre-trained weight matrix using a low-rank decomposition, which significantly reduces the number of trainable parameters.

In our research, we compared full-parameter fine-tuning and LoRA adaptation on various Hungarian large language models, specifically the PULI models, across six Hungarian language understanding benchmarks.

2. Related work

Transformers [8] underpin most large language models (LLMs) today, including Hungarian-specific models like PULI [10] and instruction-tuned variants [9]. Traditional full-parameter fine-tuning is powerful but computationally expensive. To address this, Low-Rank Adaptation (LoRA) was proposed as a parameter-efficient alternative [3]. LoRA freezes original weights and learns low-rank updates, significantly reducing resource usage.

Israel et al. [4] demonstrate LoRA's practical gains, including reduced training time and memory usage. Further analytical work by Shuttleworth et al. [7] reveals that LoRA and full fine-tuning lead to structurally different weight matrix spectra, raising concerns about long-term forgetting, especially in continual learning. Complementary approaches such as adaptive budget allocation [11] and few-shot tuning [6] also explore PEFT strategies.

In the Hungarian context, the HuLU benchmark [2, 5] offers a standardized testbed for evaluating LLMs. Our study builds on this infrastructure by comparing LoRA and full fine-tuning methods on PULI models across HuLU tasks.

3. Methods and experiments

Full fine-tuning involves updating all weights of the pre-trained model $\theta \in \mathbb{R}^{n \times m}$, requiring high memory and compute resources. In contrast, LoRA [3] introduces low-rank matrices $A \in \mathbb{R}^{n \times r}$ and $B \in \mathbb{R}^{r \times m}$, such that weight updates are expressed as $\Delta W = AB$ with $r \ll \min(n, m)$. During training, the original weights remain frozen, and only A and B are updated. This significantly reduces the number of trainable parameters and memory usage.

Practically, LoRA is applied to the attention and feed-forward linear projections in transformer blocks. We follow the common configuration of inserting LoRA into the query and value matrices, using rank values between 8 and 16. This structure allows LoRA to approximate full-rank updates while retaining efficiency.

Our experiments were conducted on four different PULI large language models:

- PULI 3SX [10]: A monolingual Hungarian GPT-NeoX model with 6.7 billion parameters, trained on 36.3 billion Hungarian words.
- PULI Trio [10]: A Hungarian-English-Chinese trilingual GPT-NeoX model with 7.67 billion parameters, trained on over 150 billion words, including 41.5 billion Hungarian words.
- PULI LlumiX 32K [9]: A continually pre-trained Llama 2 model for Hungarian with 6.74 billion parameters, trained on 7.9 billion Hungarian words from long documents only. The context length was extended to 32K tokens.
- PULI-LlumiX-Llama-3.1[1]: A continually pre-trained Llama 3.1 Instruct model for Hungarian with 8.03 billion parameters, trained on 8.08 billion Hungarian words, concluding with a Hungarian-only dataset.

For evaluation, we used six Hungarian HuLU benchmarks [5]:

- Hungarian CommitmentBank Corpus (HuCB): The HuCommitmentBank is a collection of short texts where at least one sentence includes a subordinate clause under an inference-cancelling operator. The premise is the full text; the hypothesis is the embedded clause. The task is to assess the author's level of commitment to the truth of the subordinate clause.
- Hungarian Corpus of Linguistic Acceptability (HuCOLA): The corpus contains Hungarian sentences labeled for acceptability (0/1), collected from three linguistic books.
- Hungarian Choice of Plausible Alternatives Corpus (HuCOPA): The dataset contains instances, each with a premise and two alternatives. The task is to choose the alternative that is causally related to the premise.
- Hungarian Recognizing Textual Entailment dataset (HuRTE): The dataset contains instances, each with a premise (sometimes multi-sentence) and a one-sentence hypothesis. The task is to determine whether the premise entails the hypothesis.
- Hungarian version of the Stanford Sentiment Treebank (HuSST): The dataset contains sentences, each labeled for sentiment on a three-point scale (negative, neutral, positive).

[1] https://huggingface.co/NYTK/PULI-LlumiX-Llama-3.1

- Anaphora resolution datasets for Hungarian as an inference task (HuWNLI): This Hungarian anaphora resolution dataset is framed as a sentence pair classification task. Each pair is created by replacing an ambiguous pronoun with possible referents, and the task is to determine which version is correct.

For the HuCB, HuCOLA, HuRTE, HuSST, and HuWNLI benchmarks, we trained the models using a sequence classification setup, while for HuCoPA, we trained the models as a multiple-choice task. For tasks with multiple fields, such as HuRTE with a *premise* and a *hypothesis*, we concatenated them using the *[SEP]* separator token to fit classification tasks that require a single text and label field.

To perform the fine-tuning experiments, we used the Hugging Face implementation for full-parameter fine-tuning[2]. In these experiments, it was necessary to configure both the Accelerate [1] and FSDP [12] methodologies; otherwise, training resulted in out-of-memory errors, even when using four A100 GPUs (80GB each). Despite using Accelerate and FSDP, full-parameter fine-tuning still required at least two GPUs to run successfully.

For the LoRA experiments, we used the HuLU-evaluate library's implementation [2]. For this task, a single GPU was sufficient. The following LoRA hyperparameters were used: r = 8, LoRA alpha = 32; LoRA dropout = 0.1. Since neither the Hugging Face implementation[3] of the GPT-NeoX nor the LLaMA models supports the multiple-choice task type, we implemented this functionality ourselves, based on the HuLU-evaluate framework.

4. Results

Table 1 presents a comparative evaluation of the four Hungarian PULI models (3SX, Trio, LlumiX, and LlumiX-Llama-3.1) on the six HuLU tasks (HuCB, HuCOLA, HuCoPA, HuRTE, HuSST, HuWNLI), using either full fine-tuning or parameter-efficient LoRA adaptation. Results are reported using task-specific metrics, consistent with the original HuLU benchmarks[4]: F1 score (F1), matthews correlation (MCC), and accuracy (ACC).

Overall, LlumiX-Llama-3.1 consistently delivers the strongest results, especially when fine-tuned with LoRA. It achieves state-of-the-art (SOTA) scores on five out of six benchmarks, notably reaching 74.4 F1 on HuCB, 71.9 MCC on HuRTE, and 73.1 ACC on HuWNLI. In addition, on HuCOLA, LlumiX-Llama-3.1 also achieves a SOTA result, but with full-parameter fine-tuning.

Interestingly, LoRA fine-tuned models often outperform their fully fine-tuned counterparts — particularly visible in the case of HuRTE and HuWNLI tasks, suggesting that LoRA not only reduces training cost but may also act as a regularizer, improving generalization.

Performance varies strongly between models and tasks. While 3SX performs competitively on HuSST and HuCB, it lags behind on reasoning-heavy tasks like

[2] https://github.com/huggingface/transformers/tree/main/examples/pytorch
[3] https://github.com/huggingface/transformers/tree/main/src/transformers/models
[4] https://hulu.nytud.hu

Table 1. Full-parameter and LoRA results.

		PULI 3SX	PULI Trio	PULI LlumiX	PULI-LlumiX-Llama-3.1
HuCB (F1)	Full	**62.0**	55.6	**65.9**	66.0
	LoRA	60.2	**58.2**	64.1	**74.4**
HuCOLA (MCC)	Full	**59.5**	60.8	64.1	**71.0**
	LoRA	59.0	**63.1**	**70.3**	69.2
HuCoPA (MCC)	Full	3.7	31.9	**73.4**	73.2
	LoRA	**5.3**	**44.5**	64.2	**74.1**
HuRTE (MCC)	Full	44.7	44.6	52.6	61.1
	LoRA	**55.5**	**59.2**	**68.2**	**71.9**
HuSST (ACC)	Full	**79.7**	78.5	80.2	81.5
	LoRA	79.3	**79.1**	**81.9**	**82.2**
HuWNLI (ACC)	Full	51.5	63.4	59.7	59.7
	LoRA	**58.2**	**64.9**	**67.2**	**73.1**

HuCoPA and HuRTE. In the case of Trio, LoRA consistently outperformed full fine-tuning, showing moderate improvements, although it still did not reach LlumiX-level performance. LlumiX demonstrates robust and stable results across all tasks, showing that scaling up model capacity leads to significant gains in Hungarian natural language understanding benchmarks.

In conclusion, these results validate the use of LoRA fine-tuning with larger, task-adapted PULI models and underscore the potential of LlumiX-based architectures for Hungarian language tasks.

The superior performance of LoRA, particularly on tasks with limited training data, can be partly attributed to its regularization effect. For instance, the HuCOPA benchmark contains only 400 training samples, making the full-parameter model highly susceptible to overfitting due to its significantly larger number of trainable parameters. In this constrained data setting, LoRA's reduced parameter space naturally constrains the model's capacity, thereby improving generalization. This effect is less pronounced for larger datasets like HuCOLA (7276 training samples), where the risk of overfitting is lower. This pattern supports the interpretation that LoRA not only enhances efficiency but also acts as an implicit regularizer, which is especially beneficial for smaller training corpora.

5. Conclusion

We compared full-parameter fine-tuning and parameter-efficient LoRA adaptation across six Hungarian natural language understanding benchmarks, using four PULI large language models. The goal was to assess whether LoRA can match or exceed

full fine-tuning performance while reducing computational cost.

Our results show that LoRA not only reduces resource requirements but often outperforms full fine-tuning, especially on larger models like LlumiX-Llama-3.1, which achieved state-of-the-art scores on most tasks. However, full-parameter fine-tuning remains competitive on select benchmarks (e.g., HuCOLA, HuCoPA).

Overall, using LoRA to adapt large models offers an effective and efficient solution for Hungarian language tasks, combining high performance with scalability.

References

[1] 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.

[2] P. Hatvani, K. Varga, Z. G. Yang: *Evaluation Library for the Hungarian Language Understanding Benchmark (HuLU)*, in: Proceedings of the 21th Hungarian Computational Linguistics Conference, Hungary: University of Szeged, Institute of Informatics, 2024.

[3] 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.

[4] A. Israel, D. Izenyi, L. Dilli: *Efficiently Fine-tuning Large Language Model: LoRA Approach* (May 2024), doi: 10.5281/zenodo.11312792.

[5] 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/.

[6] H. Liu, D. Tam, M. Mohammed, J. Mohta, T. Huang, M. Bansal, C. Raffel: *Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning*, in: Advances in Neural Information Processing Systems, ed. by A. H. Oh, A. Agarwal, D. Belgrave, K. Cho, 2022, url: https://openreview.net/forum?id=rBCvMG-JsPd.

[7] R. Shuttleworth, J. Andreas, A. Torralba, P. Sharma: *LoRA vs Full Fine-tuning: An Illusion of Equivalence*, MIT CSAIL, 2025, arXiv: 2410.21228 [cs.LG], url: https://arxiv.org/abs/2410.21228.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin: *Attention is All you Need*, 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. 5998–6008.

[9] 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 Instruct-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.

[10] 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.

[11] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, T. Zhao: *Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning*, in: The Eleventh International Conference on Learning Representations, 2023, url: https://openreview.net/forum?id=lq62uWRJjiY.

[12] 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 [cs.DC], URL: https://arxiv.org/abs/2304.11277.

Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 233–242
DOI: 10.17048/fmfai.2025.233

Under the hood
An inside look at PULI models*

Zijian Győző Yang[a], **Lili Anna Stajer**[b], **Gergely Lukács**[b]

[a]ELTE Research Centre for Linguistics
yang.zijian.gyozo@nytud.elte.hu

[b]Pázmány Péter Catholic University
Faculty of Information Technology and Bionics
lukacs@itk.ppke.hu

Abstract. Understanding the internal structure and behavior of large language models remains a key challenge in natural language processing. In this work, we present a comprehensive analysis of the PULI family of Hungarian generative large language models. Our study combines static analysis of model parameters with dynamic visualization of model behavior during inference. The static analysis reveals patterns in parameter distributions and dimensionality across layers, offering insight into how different layers specialize. The dynamic analysis integrates an adapted version of BertViz into a web-based interface that enables interactive exploration of attention mechanisms for arbitrary prompts and generated responses. This dual approach advances interpretability and facilitates further research on the internal mechanics of transformer models tailored for low-resource languages like Hungarian.

Keywords: PULI models, large language models, transformers visualization, attention analysis, BertViz, principal component analysis, cumulative explained variance

AMS Subject Classification: 68T07, 68T30, 68T50, 62R07, 62R40

1. Motivation

The transformer architecture and large language models (LLMs) led to a new era in natural language processing (NLP) and, more broadly, in computer science.

*The study was funded by the National Research, the Development and Innovation Office in Hungary (RRF-2.3.1-21-2022-00004).

Although their high-level designs are well documented and such models can be trained – given sufficient data, computational resources, and expertise – the internal workings of these models remain poorly understood. Specifically, the structure and geometry of billions to trillions of parameters, organized in large matrices, present a significant challenge to interpretation.

A deeper understanding of these internal mechanisms could lead to improved overall performance, enhanced capabilities in specialized tasks (e.g., disambiguation, humor detection, or handling harmful speech), and potential simplifications of model architecture. Each of these areas represents current limitations or open challenges in existing models.

For Hungarian, the PULI family [15, 16] represents the state-of-the-art in generative large language models, including both GPT-NeoX [3] and LLaMA-based [5, 12] architectures.

In our research, we performed both static and dynamic analyses of the internal parameters of the model. The static analysis focused on examining various properties and features of the trained models. In the dynamic analysis, we enabled visualization of the model's internal representations for arbitrary input text by adopting and integrating the BertViz application [13] into our demonstration platform[1].

2. Related work

A growing body of research has focused on analyzing the internal parameter values of deep neural networks and transformer-based models. It has long been recognized that deep neural networks are capable of acquiring and encoding aspects of human semantic knowledge [11]. Investigations of the BERT model have shown that distinct subspaces within the parameter space correspond to syntactic and semantic information [10]. Additionally, different senses of a word can be distinguished and separated in this space. Further studies have uncovered links between vector geometries and syntactic structures such as parse trees [6]. Recent research has also demonstrated that attributes like textual toxicity can be identified by analyzing internal parameters [8]. Regarding training dynamics, it has been found that low-dimensional structures within the parameter space are critical for enabling efficient optimization and successful model training [9]. Such findings lay the foundation for developing improved, faster, and more resource-efficient learning strategies [2].

Multilingual BERT models, when analyzed through morphosyntactic probing, have yielded further insights – for instance, indicating that preceding context often contains more semantically relevant information than the following context [1]. Model compression, particularly through quantization, is another active research area with significant practical implications [4].

In parallel with analytical approaches, there have been efforts to improve the interpretability of semantic and contextual representations in LLMs during infer-

[1] https://juniper.nytud.hu/demo/visualizer

ence. Tools such as ExBERT, a visualization framework for exploring learned representations in transformer models [7], and BertViz, a multiscale visualization tool applicable to any transformer architecture, have proven useful in this regard. BertViz has been employed, for example, to detect bias and trace specific behaviors back to particular model components [13, 14].

3. Static analysis

3.1. PULI LlumiX 32K model and its parameters

The static analysis in this study was conducted on the PULI LlumiX 32K model [15], which is based on LLaMA-2-7B-32K[2] variant of the open-source LLaMA (Large Language Model Meta AI) 2 family [12] introduced by Meta[3] in 2023. LLaMA models are decoder-only architectures, meaning they consist solely of transformer decoder layers. The core architecture comprises multiple identical layers, each containing a feed-forward neural network (FFN), layer normalization, and self-attention blocks. Input data is processed through an embedding layer and positional encoding before being passed through the stacked layers.

The self-attention mechanism maps a query and a set of key–value pairs to an output, enabling the model to capture dependencies between tokens regardless of their position in the sequence. The main components and parameters involved are as follows:

- *Query* (Q): Represents the current token being processed and is used to compute attention scores by comparing it to all other tokens' key vectors.
- *Key* (K): Associated with each token in the sequence and used to determine the relevance of other tokens to the current one.
- *Value* (V): Also associated with each token, and contains the information that contributes to the final weighted output.

The result of the self-attention mechanism is a weighted sum of the value vectors, where weights are derived from the similarity between queries and keys. Modern transformer models use multi-head attention, which involves multiple parallel self-attention mechanisms, each with its own set of learned parameters. Dedicated weight matrices are used to compute the Q, K, and V vectors from the input representations.

The LLaMA-2-7B-32K model consists of 32 transformer layers and approximately 7 billion parameters. A detailed breakdown of the model's architecture, including the dimensionality of parameter matrices and the total parameter count, is provided in Table 1.

[2] https://huggingface.co/togethercomputer/LLaMA-2-7B-32K
[3] https://www.meta.ai

Table 1. Number of Parameters in LLaMA-2-7B-32K.

Description	**Matrix size**	**Count**	**Parameter count**
embed_token_weight: maps each token onto the d_model	(32000,4096)	1	131 072 000
input_layernorm: each layer input normalized	(4096,1)	32	131 072
self_attn_k: multi-attention head W_K matrix	(4096,4096)	32	536 870 912
self_attn_q: multi-attention head W_Q matrix	(4096,4096)	32	536 870 912
self_attn_v: multi-attention head W_V matrix	(4096,4096)	32	536 870 912
self_attn_o: multi-attention head W_O matrix	(4096,4096)	32	536 870 912
post_attention_layernorm: each multi-head attention output normalized	(4096, 1)	32	131 072
mlp.down_proj: FNN weights	(4096,11008)	32	1 442 840 576
mlp.gate_proj: FNN gate weights	(11008,4096)	32	1 442 840 576
mlp.up_proj: FNN weights	(11008,4096)	32	1 442 840 576
norm: normalizing function for last layer output	(4096,1)	1	4096
lm_head: maps d_model back onto the vocabulary space	(32000,4096)	1	131 072 000
TOTAL			6 738 415 616

3.2. Analysis and results

Distinct patterns were observed in the model's parameters. In the feedforward network (FNN), the standard deviation of the down, gate, and up projection weights progressively increases in the upper layers (Figure 1). This trend is further supported by a decrease in the 25 percentile and an increase in the 75 percentile values (Figure 2). Notably, the first and last layers exhibit substantially larger changes compared to the intermediate layers.

In the self-attention blocks, the standard deviations of the key (k) and query (q) weights decrease from the lower to the upper layers, while the value (v) and output (o) weights show a similar downward trend. Again, notable exceptions to these

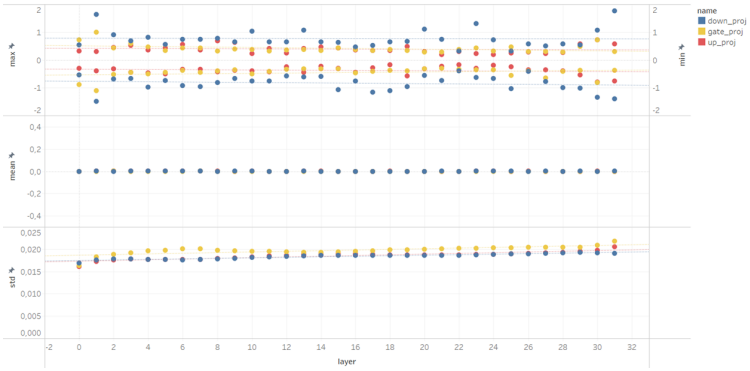


Figure 1. Plot of min-max, mean and standard deviation value of FNN components.

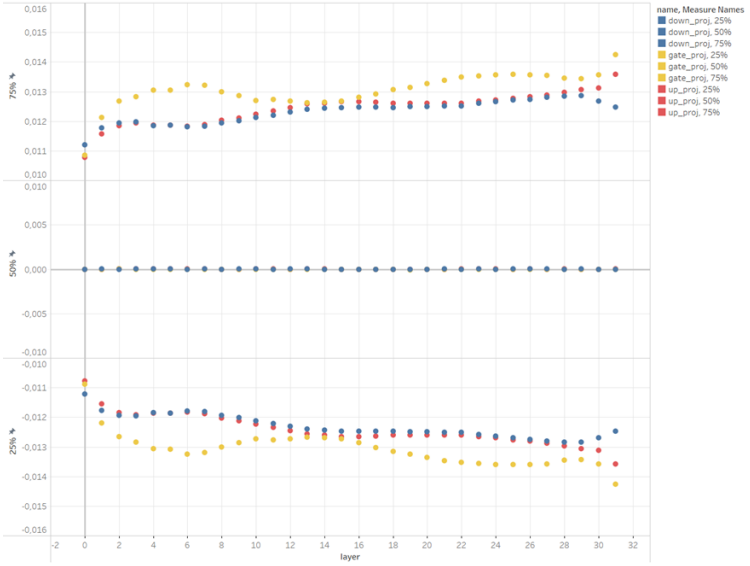


Figure 2. Plot of 25, 50 and 75 percentile values of FNN components.

trends appear in the first and last layers. Principal component analysis on these weights revealed that, in general, the cumulative explained variance indicates that dimensionality cannot be significantly reduced without information loss. However, in a few specific cases – particularly for the k and q weights, and to a lesser extent the v and o weights – early layers (especially the first three) exhibit high cumulative explained variance, approaching 1, with a substantially smaller number of dimensions than in later layers (Figure 3, Figure 4).

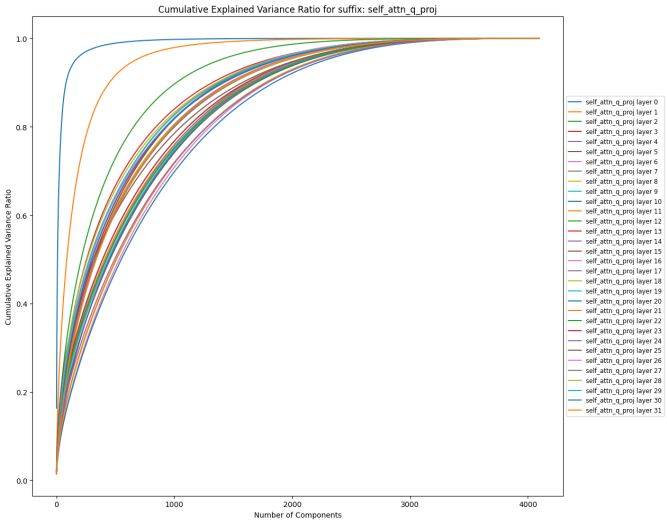


Figure 3. Cumulative explained variance of query (q) matrices for self-attention layers.

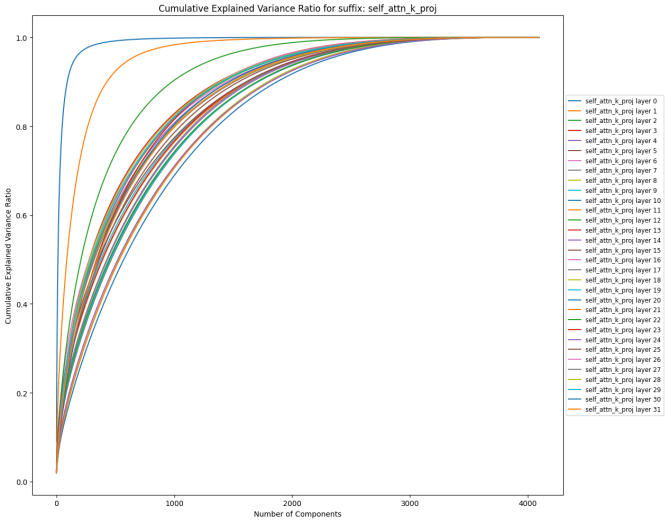


Figure 4. Cumulative explained variance of key (k) matrices for self-attention layers.

4. Dynamic analysis

For Dynamic analysis, we integrated the BertViz tool [13] into our demo site. In our demo site[4], we split the BertViz output into frontend and backend components and integrated them into the corresponding sections of our site. While the original BertVis frontend code remained unchanged, we applied several modifications to the backend. First, we added a text generation module and then merged the newly generated text with the original input prompt. This functionality allows us to observe the relationships between the prompt and its response.

Figure 5 presents the architecture of our dynamic analysis demo site. This architecture diagram illustrates a system designed to interface with a LLM through a frontend-backend pipeline. On the frontend, users interact with the system via a Demo interface, where they input prompts. These prompts are sent to the LLM hosted in our backend, which generates corresponding model responses and returns them to the frontend for display. Simultaneously, a weight extraction module accesses internal data (such as attention weights) from the LLM, processes it, and forwards the resulting weights to BertViz, a frontend visualization tool that allows users to explore the model's inner workings. This design separates user interaction, model computation, and interpretability, enabling a clear and interactive workflow to use and understand the behavior of the LLM.

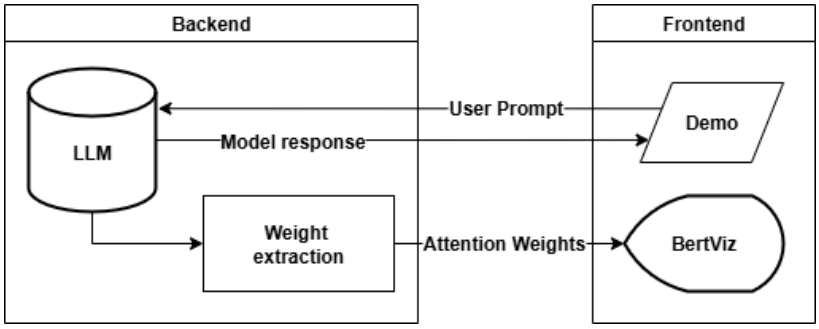
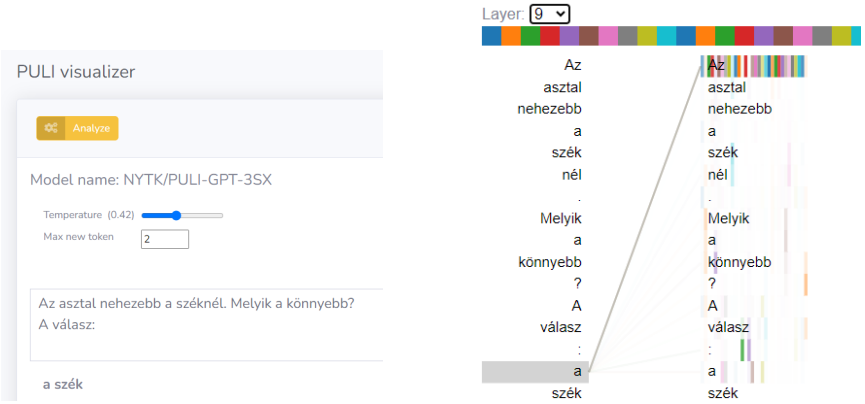


Figure 5. Architecture of the demo site.

In Figure 6, we show the integrated BertViz visualization along with the dynamic prompt-response analysis. In our demo site, an input prompt can be provided, and then the response will be generated (see Figure 6a). In this example, the prompt is: *The table is heavier than the chair. Which is the lighter one? The answer:*. The response: *the chair*. In Figure 6b, we visualize the relationships between the prompt and its response. In this example, we observe the weights of layer 9, where we can see that, in the case of 'a' (*the*), the attention is focused more on the word 'szék' (*chair*) than on the word 'asztal' (*table*).

[4] https://juniper.nytud.hu/demo/visualizer



(a) A screenshot of our demo site.

(b) Illustrating the relationship between the input prompt and the generated response.

Figure 6. The PULI visualizer demo.

5. Conclusion

In this paper, we investigated the internal mechanisms of the PULI large language models using a two-pronged approach: static parameter analysis and dynamic behavior visualization. Our static examination highlighted systematic trends in weight distributions and dimensionality across layers, suggesting layer-specific roles in the model's computation. The dynamic component extended the BertViz framework, allowing users to explore the relationship between input prompts and model responses in real time. These findings contribute to the broader goal of demystifying LLMs and open avenues for improving model transparency, fine-tuning strategies, and error diagnosis, particularly in the context of Hungarian language technologies. Future work may focus on extending these methods to multilingual settings or applying similar techniques to fine-tuning and alignment tasks.

In the future, we plan to extend our experiments to other PULI models and implement a model selection module that allows users to interactively switch between different PULI architectures, including encoder-only, decoder-only, and encoder-decoder models. This would enable comparative analysis, generalization of results, and adaptive usage based on specific task requirements. In addition, combining advanced statistical methods with visualizations appears promising for dynamic analysis.

References

[1] J. Acs, E. Hamerlik, R. Schwartz, N. A. Smith, A. Kornai: *Morphosyntactic probing of multilingual BERT models*, Natural Language Engineering 30.4 (2024), pp. 753–792, doi: 10.1017/S1351324923000190.

[2] G. Berend: *Masked Latent Semantic Modeling: an Efficient Pre-training Alternative to Masked Language Modeling*, in: Findings of the Association for Computational Linguistics: ACL 2023, ed. by A. Rogers, J. Boyd-Graber, N. Okazaki, Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 13949–13962, doi: 10.18653/v1/2023.findings-acl.876, url: https://aclanthology.org/2023.findings-acl.876/.

[3] 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, url: https://aclanthology.org/2022.bigscience-1.9/.

[4] T. Dettmers, R. A. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, D. Alistarh: *SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression*, in: The Twelfth International Conference on Learning Representations, 2024, url: https://openreview.net/forum?id=Q1u25ahSuy.

[5] A. Grattafiori et al.: *The Llama 3 Herd of Models*, 2024, arXiv: 2407.21783 [cs.AI], url: https://arxiv.org/abs/2407.21783.

[6] J. Hewitt, C. D. Manning: *A Structural Probe for Finding Syntax in Word Representations*, 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. 4129–4138, doi: 10.18653/v1/N19-1419, url: https://aclanthology.org/N19-1419/.

[7] B. Hoover, H. Strobelt, S. Gehrmann: *exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models*, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ed. by A. Celikyilmaz, T.-H. Wen, Online: Association for Computational Linguistics, July 2020, pp. 187–196, doi: 10.18653/v1/2020.acl-demos.22, url: https://aclanthology.org/2020.acl-demos.22/.

[8] A. Lee, X. Bai, I. Pres, M. Wattenberg, J. K. Kummerfeld, R. Mihalcea: *A Mechanistic Understanding of Alignment Algorithms: A Case Study on DPO and Toxicity*, in: Proceedings of the 41st International Conference on Machine Learning, ed. by R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, F. Berkenkamp, vol. 235, Proceedings of Machine Learning Research, PMLR, 21–27 Jul 2024, pp. 26361–26378, url: https://proceedings.mlr.press/v235/lee24a.html.

[9] J. Mao, I. Griniasty, H. K. Teoh, R. Ramesh, R. Yang, M. K. Transtrum, J. P. Sethna, P. Chaudhari: *The training process of many deep networks explores the same low-dimensional manifold*, Proceedings of the National Academy of Sciences 121.12 (2024), e2310002121, doi: 10.1073/pnas.2310002121.

[10] E. Reif, A. Yuan, M. Wattenberg, F. B. Viégas, A. Coenen, A. Pearce, B. Kim: *Visualizing and Measuring the Geometry of BERT*, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, R. Garnett, 2019, pp. 8592–8600, url: https://proceedings.neurips.cc/paper/2019/hash/159c1ffe5b61b41b3c4d8f4c2150f6c4-Abstract.html.

[11] A. M. Saxe, J. L. McClelland, S. Ganguli: *A mathematical theory of semantic development in deep neural networks*, Proceedings of the National Academy of Sciences 116.23 (2019), pp. 11537–11546, doi: 10.1073/pnas.1820226116.

[12] H. Touvron et al.: *Llama 2: Open Foundation and Fine-Tuned Chat Models* (2023), arXiv: 2307.09288 [cs.CL].

[13] J. Vig: *A Multiscale Visualization of Attention in the Transformer Model*, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ed. by M. R. Costa-jussà, E. Alfonseca, Florence, Italy: Association for Computational Linguistics, July 2019, pp. 37–42, doi: 10.18653/v1/P19-3007, url: https://aclanthology.org/P19-3007/.

[14] J. Vig, Y. Belinkov: *Analyzing the Structure of Attention in a Transformer Language Model*, in: Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, ed. by T. Linzen, G. Chrupała, Y. Belinkov, D. Hupkes, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 63–76, doi: 10.18653/v1/W19-4808, url: https://aclanthology.org/W19-4808/.

[15] 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 Instruct-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.

[16] 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.