Annales Mathematicae et Informaticae
58 (2023) pp. 81–91
DOI: 10.33039/ami.2023.11.002
URL: https://ami.uni-eszterhazy.hu

CAPTCHA recognition using machine learning algorithms with various techniques

Ádám Kovács[ab], **Tibor Tajti**[a]

[a]Eszterházy Károly Catholic University
kovacs2.adam@uni-eszterhazy.hu
tajti.tibor@uni-eszterhazy.hu

[b]University of Debrecen, Doctoral School of Informatics

Abstract. In this paper, we present research results on the recognition of text-based CAPTCHA tests using advanced machine learning algorithms and techniques. Text-based CAPTCHAs serve as a crucial security measure to prevent automated access to various web services, but their effectiveness depends on their resistance to sophisticated recognition techniques. To this end, we focus on evaluating and enhancing the performance of recognition models using a Convolutional Neural Network (CNN) as the base model. We propose an integrated approach, which incorporates a systematic parameter optimization strategy using Grid Search Cross-Validation (Grid Search CV) and the Ensemble Voting Method to improve the performance of the recognition model. The use of Grid Search CV enables us to fine-tune the hyperparameters of the CNN model, leading to an optimal configuration. Further, we investigate the effectiveness of the Ensemble Voting Method to aggregate the predictions from multiple CNN models, each with a set of the optimal parameters obtained from the Grid Search CV. The methods' performance was evaluated through multiple learning sessions, assessing their effectiveness in recognizing text-based CAPTCHAs under various scenarios.

Keywords: Machine learning, CAPTCHA recognition, neural networks, hyperparameter optimization, ensemble methods

1. Introduction

CAPTCHA, or Completely Automated Public Turing Test to tell Computers and Humans Apart, is a widely used security measure designed to differentiate between

Submitted: August 31, 2023
Accepted: November 9, 2023
Published online: November 10, 2023

human and machine users [18]. However, with recent advancements in artificial intelligence, traditional CAPTCHAs are becoming increasingly susceptible to automated system bypassing.

Convolutional Neural Networks (CNNs) are a category of deep learning algorithms that are generally used for processing and analyzing visual data, such as images and videos. Their distinctive architecture leverages spatial hierarchies and local patterns within the data, enabling the automatic learning of complex and abstract features. While CNNs are highly applicable to a range of computer vision tasks, including image recognition, object detection, and segmentation, they can also be employed in other domains, such as time series prediction and speech recognition. The use of CNNs to recognize distorted characters has exposed the vulnerability of existing CAPTCHA systems, emphasizing the need for more sophisticated and resilient alternatives [7].

Grid Search Cross-Validation (Grid Search CV) is a hyperparameter optimization technique in machine learning models [9]. The use of Grid Search CV entails a comprehensive search across a defined range of hyperparameter values, with the performance of each combination assessed via cross-validation (CV). This approach aids in determining the optimal set of hyperparameters, resulting in superior model performance. Grid Search CV is crucial for developing robust models, as it ensures that they are fine-tuned and capable of generalizing effectively to unseen data.

Ensemble methods comprise a collection of powerful machine learning techniques that focus on integrating multiple models to achieve enhanced predictive performance compared to individual models. The core concept underlying ensemble methods are to exploit diversity among various models, which assists in reducing prediction errors, increasing stability, and bolstering generalization capabilities. By aggregating the predictions of several models, ensemble methods can counterbalance the limitations of individual models, ultimately yielding more accurate and robust predictions. Ensemble voting can be effective using learners with the same model [13], but also using various models for the voters [6].

1.1. Dataset

Figure 1 illustrates two text-based CAPTCHAs from the dataset, each subjected to different noise levels and distortions.

Figure 1. Random elements of the dataset.

These alterations serve to increase the CAPTCHAs' complexity for automated systems, consequently augmenting the security of the protected system. The presence of diverse distortions and noise levels in the dataset challenges the automated systems to adapt and recognize characters under varying conditions, thereby testing their robustness and reliability in solving CAPTCHAs [3].

The dataset consists of a total of 1,070 images, predominantly in the PNG format, with a few files in the JPG format [17]. Each of the images is in grayscale, featuring five alphanumeric characters that may include both letters and numbers. The dimensions of these images are 200 pixels in width and 50 pixels in height.

1.2. Model

Figure 2 shows the pre-existing model that we utilized for text-based CAPTCHA prediction, with an input layer for 50×200 grayscale images [10].

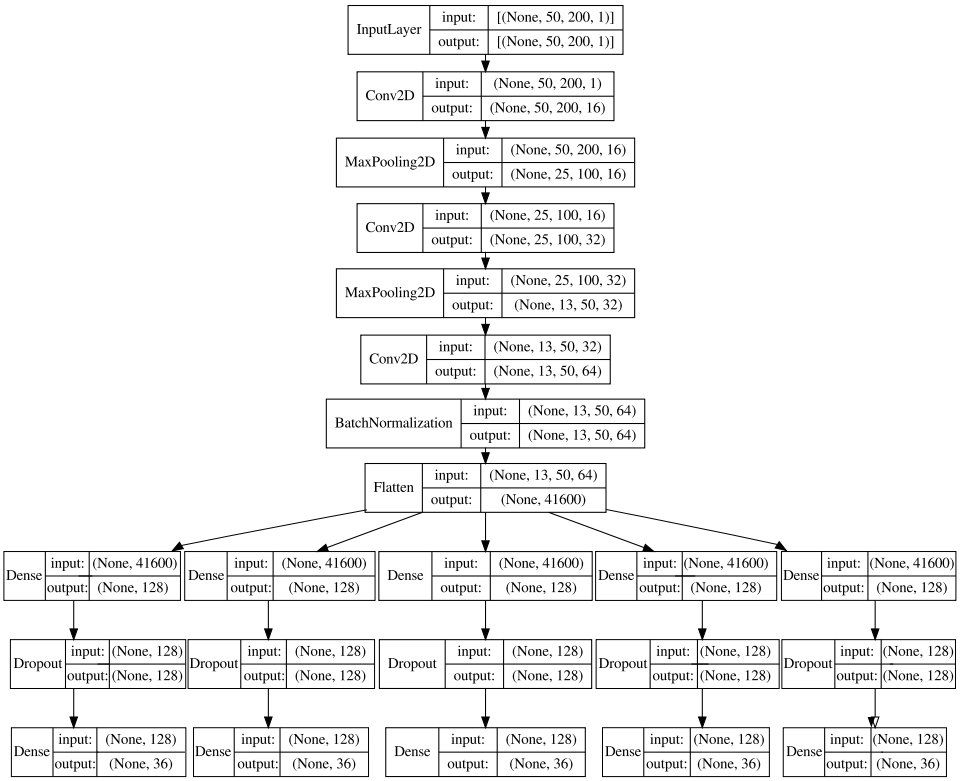

Figure 2. Representation of the layers.

It features three Conv2D layers, each followed by a MaxPooling2D layer for feature extraction, and a BatchNormalization layer for improved stability [16]. The output is flattened, and the model branches into five separate paths, each responsible for predicting one CAPTCHA character. Each branch consists of a Dense layer, a Dropout layer, and a final Dense layer with output neurons matching the number of possible characters. The activation function for the last Dense layer is the softmax function, which calculates probabilities for each possible character.

1.3. Voting method

One of the most prominent ensemble methods is the voting method [2, 5, 15]. In this approach, multiple model instances are trained on the same dataset. These trained models are then used to make predictions for new data points, and the final prediction is determined by aggregating the individual models' predictions using a voting scheme.

Various voting schemes can be employed in the voting method, such as:

- Plurality voting: The final prediction is the class (or value) that receives the most votes from the individual models.
- Fuzzy average voting: For classification tasks, the final prediction is determined by averaging the predicted class probabilities from each model and selecting the class with the highest average probability.

In practical applications, the voting method has been shown to be highly effective in a wide range of problems, such as image and speech recognition, natural language processing, and bioinformatics [1, 14]. Besides the Voting approach, other ensemble methods like Bagging and Boosting can also be applied to improve predictive performance [8].

2. Experiments and results

2.1. Performance evaluation framework

We conducted our evaluation utilizing AMD GPU in conjunction with the TensorFlow framework. Additionally, we employed the Keras library alongside TensorFlow to facilitate more straightforward and rapid implementation of neural networks.

To address the stochastic nature of the algorithms and potential discrepancies across learning sessions, we employed the k-fold Cross Validation method. We divided the dataset into 10 equal-sized subsets, and for each learning session, one fold served as the validation set while the other nine were used for training. In each learning session, every model with different parameter combinations was run 10 times and then the average performance was calculated across the 10 validation sets. To further enhance the reliability of our results, we repeated the entire procedure ten times, each time using a different random seed to shuffle the dataset before dividing it into folds. This generated a total of 100 validation sets (10 folds $\times$ 10 repetitions), and we tested all parameter combinations on these sets.

This rigorous validation technique ensured scientifically accurate and statistically reliable results, minimizing random variations and providing a solid basis for our conclusions.

For the analysis, we leveraged the Python-based Numpy and Pandas libraries to handle and manipulate the data. In addition to these libraries, we employed the matplotlib library for data visualization purposes, enabling a more comprehensive

understanding of the model's performance and facilitating the evaluation of the results.

Throughout our experiments, we utilized a CNN with the optimal set of parameters to achieve the best possible performance. Initially, we conducted a thorough search for the best parameters for the CNN model, ensuring that our chosen model exhibited the highest performance. After identifying the optimal parameters, we proceeded to apply the selected model for the voting method.

In the voting method, we employed various numbers of models for prediction, which allowed us to evaluate the performance of our approach across different ensemble sizes. This strategy not only provided valuable insights into the robustness and reliability of our selected CNN model but also enabled us to identify the optimal number of models to use in our ensemble for achieving the best possible results.

2.2. Hyperparameter optimization

To ensure that our CNN model achieved the best possible performance, we conducted an extensive hyperparameter optimization process. We employed the Grid Search CV technique from the sci-kit learn library to systematically explore the hyperparameter space and identify the optimal combination of hyperparameters for our model. The following hyperparameters and their respective candidate values were included in the grid search:

- Batch size: 16, 32, 64
- Epochs: 50
- Activation function for the convolutional layers: ReLU, ReLU6, Swish
- Number of neurons used in the penultimate dense layer: 32, 64, 128
- Activation function for the first output layer: ReLU, ReLU6, Swish
- Dropout rate: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
- Activation function for the second output layer: Softmax
- Optimizer: Adam

The grid search was conducted with 10-fold CV to ensure that the selected hyperparameters were robust and generalizable across different subsets of the dataset. This approach provided a reliable estimate of the model's performance and reduced the risk of overfitting.

Table 1 shows the best performance with the Rectified Linear Unit (ReLU) and Rectified Linear Unit 6 (ReLU6) activation functions. Thus, we focused on them, excluding the Swish function due to its lower performance. Based on this, we chose the optimal hyperparameters for our CNN model, which guided the experiments and ensemble methods.

Table 1. The 5 best performing models with varying parameter combinations using Grid Search CV.

Function	Batch Size	Dropout Rate	Units	Mean Test Score
ReLU6	64	0.5	128	0.841061
ReLU6	32	0.4	64	0.840389
ReLU	16	0.5	64	0.840010
ReLU	16	0.4	64	0.839632
ReLU	32	0.4	64	0.839001

The outcomes of our experiments with different batch sizes, neuron numbers, and dropout rates are illustrated in Figures 3, 4, and 5.

These figures provide a comprehensive overview of the mean test scores achieved with various combinations of these hyperparameters, assessed using Grid Search CV. In these results, the ReLU6 is used as an activation function for the convolutional layers and the first output layer. This choice was based on our preliminary analysis, which indicated that ReLU6 outperformed other activation functions in our specific problem setting.

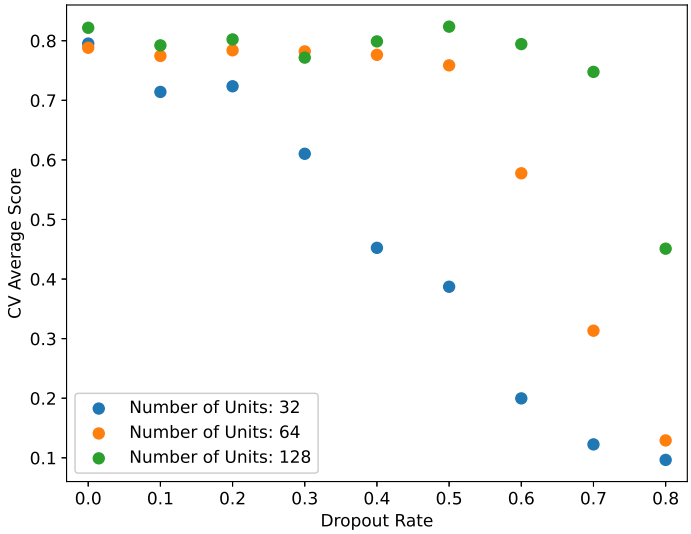

Figure 3. Mean test scores with batch size of 16 using Grid Search CV.

Figure 4 shows that as the batch size increases, the model can learn with a higher dropout rate, enabling more effective regularization and improved generalization performance. This observation is consistent with the idea that a larger batch size provides more accurate gradient estimates, allowing the model to handle the higher

levels of noise introduced by dropout.

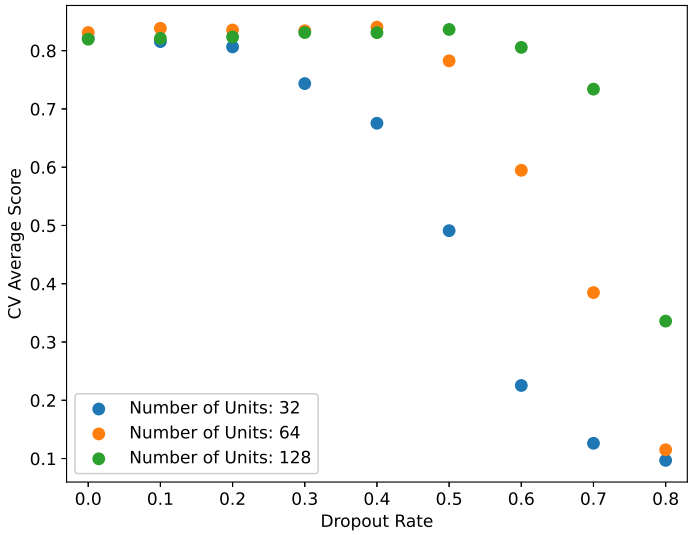

Figure 4. Mean test scores with batch size of 32 using Grid Search CV.

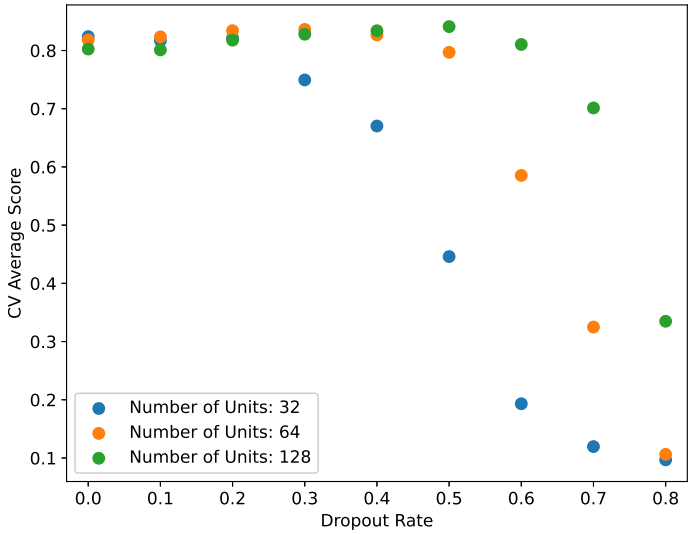

Figure 5. Mean test scores with batch size of 64 using Grid Search CV.

The higher the number of units in the model, the more effectively it can learn and utilize a higher dropout rate. This is likely because a larger number of neurons

enable the model to represent more complex functions, counterbalancing the effect of dropout.

Figure 4 and 5 show that using a dropout rate equal to or greater than 0.7 does not result in any significant improvement in the model's performance. In fact, dropout rates of 0.7 or higher may lead to degraded performance due to excessive noise in the learning process, which could hinder the model from capturing important patterns in the data.

The 0.4 and 0.5 dropout rates produced the best results, providing a good balance between introducing noise to promote generalization and maintaining sufficient signals for the model to learn the underlying patterns. The optimal neuron numbers for our model were 64 and 128. These values yielded the best results across various batch sizes and dropout rates, indicating that they provide an appropriate level of model complexity to learn from the data without overfitting.

2.3. Performance of voting functions

By employing various voting schemes to combine the predictions of various models trained on the same dataset, the ensemble approach effectively reduces errors, increases stability, and enhances generalization capabilities. It is important to note that the models involved in this ensemble approach do not differ in their architecture or parameters; the differences between them arise from the individual training processes they undergo. As previously discussed in Section 1.3, the plurality voting function selects the prediction with the highest number of votes, while the fuzzy average voting function calculates the average of all predictions.

Table 2 presents the results of an experiment conducted to evaluate the performance of two voting schemes, plurality and fuzzy average voting functions. The experiment was performed 10,000 times, with each iteration involving a varying number of models, ranging from 2 to 30. The number of models used in the experiment increased incrementally by two in each iteration, providing a detailed view of the performance trends as the ensemble size grew.

In total, 60 models were used, and for each iteration, a random selection of the required number of models was made from these models. The models' predictions were based on the same dataset.

From the analysis of Figure 6, it can be observed that there is a gradual improvement in the performance of both the plurality voting function and the fuzzy average voting function as the number of models increases. This indicates that the ensemble of models can effectively leverage the strengths of individual models to reduce errors and increase stability. The voting functions tend to perform better when there is more diversity among the models, as it allows for a more robust decision-making process.

Further analysis reveals that as the number of models increases, the achieved results also improve, indicating higher accuracy in predictions. The fuzzy average voting function consistently outperforms the plurality voting function. This advantage is more noticeable with fewer models, but beyond 14–16 models, this

Table 2. Performance comparison of plurality and fuzzy average voting functions across different numbers of models.

Models	Plurality Voting Function	Fuzzy Average Voting Function
2	0.887717	0.906253
4	0.908664	0.915228
6	0.914723	0.918222
8	0.917318	0.919663
10	0.918746	0.920627
12	0.919694	0.921418
14	0.920391	0.922023
16	0.920961	0.922385
18	0.921416	0.922765
20	0.921691	0.923070
22	0.921951	0.923260
24	0.922246	0.923641
26	0.922408	0.923862
28	0.922506	0.924067
30	0.922789	0.924160

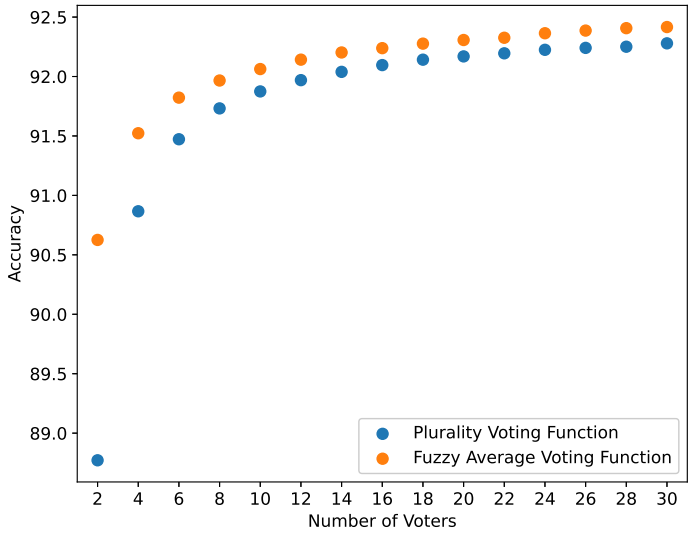

Figure 6. The performance results of voting functions by 2–30 voters on test data.

superiority stabilizes to a consistent advantage of approximately 0.001 to 0.0015 in favor of the fuzzy average voting function.

The stable advantage suggests that the fuzzy average voting function, by considering the average of predictions rather than just the most frequent one, provides more accurate results. However, from 14–16 models onwards, the rate of improvement in the results seems to decrease somewhat for both voting functions, suggesting that while incorporating more models can enhance performance, there may be diminishing returns beyond a certain point. The greater accuracy of the fuzzy average voting function may be attributed to its ability to incorporate more information from the models' outputs compared to the plurality voting function.

3. Conclusions

In conclusion, our results suggest that a careful choice of batch size, number of neurons in the penultimate dense layer, and dropout rates can significantly impact the performance of deep learning models. By employing grid search cross-validation, we were able to identify optimal combinations of these hyperparameters, leading to improved generalization and higher mean test scores. Furthermore, the experiment demonstrates that the choice of the voting scheme can have a considerable impact on the performance of an ensemble of models trained on the same dataset. While both plurality and fuzzy average voting functions can provide some benefits, the plurality voting function appears to offer more consistent improvements in performance as the number of models increases.

To further advance the field and enhance model performance, future research could explore the following developments: employing segmentation techniques to refine the input data, exploring the potential benefits of using fuzzification techniques for refining binary class membership values during model training, investigating alternative ensemble methods that may provide additional benefits, experimenting with different datasets to assess the robustness of the models, and incorporating various machine learning models, such as recurrent neural networks, to address the specific challenges of the task [4, 11, 12]. While the results presented in this study are promising, it is important to conduct additional research and analysis to fully comprehend the behavior and potential of these models, as this understanding can ultimately lead to more accurate and reliable methods.

References

[1] R. Atallah, A. Al-Mousa: *Heart Disease Detection Using Machine Learning Majority Voting Ensemble Method*, in: 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019, pp. 1–6, doi: 10.1109/ICTCS.2019.8923053.

[2] M. Brill, R. Freeman, S. Janson, M. Lackner: *Phragmén's voting methods and justified representation*, Mathematical Programming (2023), doi: 10.1007/s10107-023-01926-8.

[3] E. Bursztein, M. Martin, J. Mitchell: *Text-based CAPTCHA strengths and weaknesses*, in: Proceedings of the 18th ACM conference on Computer and communications security, 2011, pp. 125–138.

[4] J. Chen, X. Luo, Y. Liu, J. Wang, Y. Ma: *Selective Learning Confusion Class for Text-Based CAPTCHA Recognition*, IEEE Access 7 (2019), pp. 22246–22259, DOI: 10.1109/ACCESS.2019.2899044.

[5] T. G. Dietterich: *Ensemble Methods in Machine Learning*, in: Multiple Classifier Systems, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15, ISBN: 978-3-540-45014-6, DOI: 10.1007/3-540-45014-9_1.

[6] I. Fazekas, A. Barta, L. Fórián: *Ensemble noisy label detection on MNIST*, Annales Mathematicae et Informaticae 53 (2021), pp. 125–137, DOI: 10.33039/ami.2021.03.015.

[7] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al.: *Recent advances in convolutional neural networks*, Pattern Recognition 77 (2018), pp. 354–377.

[8] L. Kabari, U. Onwuka: *Comparison of Bagging and Voting Ensemble Machine Learning Algorithm as a Classifier*, International Journal of Computer Science and Software Engineering 9 (Mar. 2019), pp. 19–23.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al.: *Scikit-learn: Machine learning in Python*, the Journal of Machine Learning Research 12 (2011), pp. 2825–2830.

[10] A. Shawon: *Captcha Recognition*, Accessed: 2023-04-02, 2023, URL: https://www.kaggle.com/code/shawon10/captcha-recognition.

[11] Y. Shu, Y. Xu: *End-to-End Captcha Recognition Using Deep CNN-RNN Network*, in: 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2019, pp. 54–58, DOI: 10.1109/IMCEC46724.2019.8983895.

[12] T. Tajti: *Fuzzification of training data class membership binary values for neural network algorithms*, Annales Mathematicae et Informaticae 2020 (Oct. 2020), DOI: 10.33039/ami.2020.10.001.

[13] T. Tajti: *New voting functions for neural network algorithms*, Annales Mathematicae et Informaticae 52 (2020), DOI: 10.33039/ami.2020.10.003.

[14] E. Tasci, C. Uluturk, A. Ugur: *A voting-based ensemble deep learning method focusing on image augmentation and preprocessing variations for tuberculosis detection*, Neural Computing and Applications 33 (2021), pp. 15541–15555, DOI: 10.1007/s00521-021-06177-2.

[15] S. Wan, H. Yang: *Comparison among Methods of Ensemble Learning*, in: 2013 International Symposium on Biometrics and Security Technologies, 2013, pp. 286–290, DOI: 10.1109/ISBAST.2013.50.

[16] J. Wang, J. Qin, X. Xiang, Y. Tan, N. Pan: *CAPTCHA recognition based on deep convolutional neural network*, Mathematical Biosciences and Engineering 16.5 (2019), pp. 5851–5861, ISSN: 1551-0018, DOI: 10.3934/mbe.2019292.

[17] R. Wilhelmy, H. Rosas: *captcha dataset*, July 2013, URL: https://www.researchgate.net/publication/248380891_captcha_dataset.

[18] Y. Zhang, H. Gao, G. Pei, S. Luo, G. Chang, N. Cheng: *A Survey of Research on CAPTCHA Designing and Breaking Techniques*, in: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2019, pp. 75–84, DOI: 10.1109/TrustCom/BigDataSE.2019.00020.