IRSTI 28.23.37 https://doi.org/10.26577/jpcsit20253201

Mukhtar Zhassuzak * , **Farida Narkeshova** , **Zholdas Buribayev** , **Bazargul Matkerim**

Al-Farabi Kazakh National University, Almaty, Kazakhstan
*e-mail: zhassuzak.mukhtar@gmail.com

DEVELOPMENT OF A GENETIC ALGORITHM FOR OPTIMIZING CONVOLUTIONAL NEURAL NETWORKS IN ORDER TO IMPROVE THE ACCURACY OF OBJECT DETECTION IN DIFFICULT LIGHTING AND BACKGROUND CONDITIONS

Abstract. This article addresses the challenge of improving object detection accuracy in video data captured under low-light conditions. Modern video detection systems–particularly in areas such as security, autonomous systems, and medicine–often suffer from reduced accuracy due to poor lighting. The proposed method is based on the integration of the YOLOv5 object detection model with a variety of image processing filters (including CLAHE, gamma correction, histogram equalization, Gaussian blur, bilateral filtering, the Non-Local Means algorithm, Gray-World and Max-RGB balancing schemes, as well as Retinex and MSRCR methods) and genetic algorithms. This approach enhances both the reliability of detection and computational efficiency. Experimental evaluations demonstrate that the proposed system achieves significantly higher object detection accuracy in low-light data compared to traditional methods.

Key words: low-light images, object detection, image enhancement, preprocessing methods, CLAHE, gamma correction, histogram equalization, noise reduction, contrast enhancement.

1. Introduction

The quality of video captured under low-light conditions significantly deteriorates, negatively impacting the accuracy of object detection algorithms [1]. Low contrast, high noise levels, and uneven light distribution in images hinder informed decision-making in domains such as security, video surveillance systems, medical diagnostics, and autonomous control [2]. Although modern approaches–particularly those based on convolutional neural networks (CNNs)–demonstrate high effectiveness in object detection, poor-quality input data in low-light environments reduces the overall performance of such systems [3].

Studies have shown that preprocessing techniques such as CLAHE, gamma correction, histogram equalization, Gaussian blur, bilateral filtering, as well as Retinex and MSRCR methods, can improve image quality, enhance contrast, and reduce noise [4],[5]. However, selecting optimal parameters to ensure the efficiency of these methods remains a challenging task. To address this issue, genetic algorithms are employed for automated parameter optimization [6]. Implemented using the DEAP library, genetic algorithms are based on principles of natural evolution, allowing for automatic tuning of filter parameters and improved YOLOv5 detection performance [6]. Due to the high computational demand, the use of parallel computing methods is crucial. Python's multiprocessing module enables the utilization of multiple CPU cores, significantly reducing the algorithm’s execution time.

This project proposes the development of an integrated system combining the YOLOv5 video detection model, preprocessing techniques, and genetic algorithms to improve the accuracy and reliability of object detection in images captured under low-light conditions. The main objectives of the project include enhancing image quality through preprocessing, implementing real-time detection with YOLOv5, automatically optimizing filter parameters using genetic algorithms, and introducing parallel computing to improve the overall system performance [1],[4],[6],[8].

2. Methods and Materials

This section provides a detailed overview of the main research methods and materials used in the study. Throughout the project, the PyTorch deep

learning library was employed to train neural networks and configure the YOLOv5 model [17]. The project focuses on enhancing the accuracy of object detection in video data captured under low-light conditions. To achieve this objective, an integrated system was implemented, combining preprocessing techniques, the YOLOv5 detection model, and genetic algorithms.

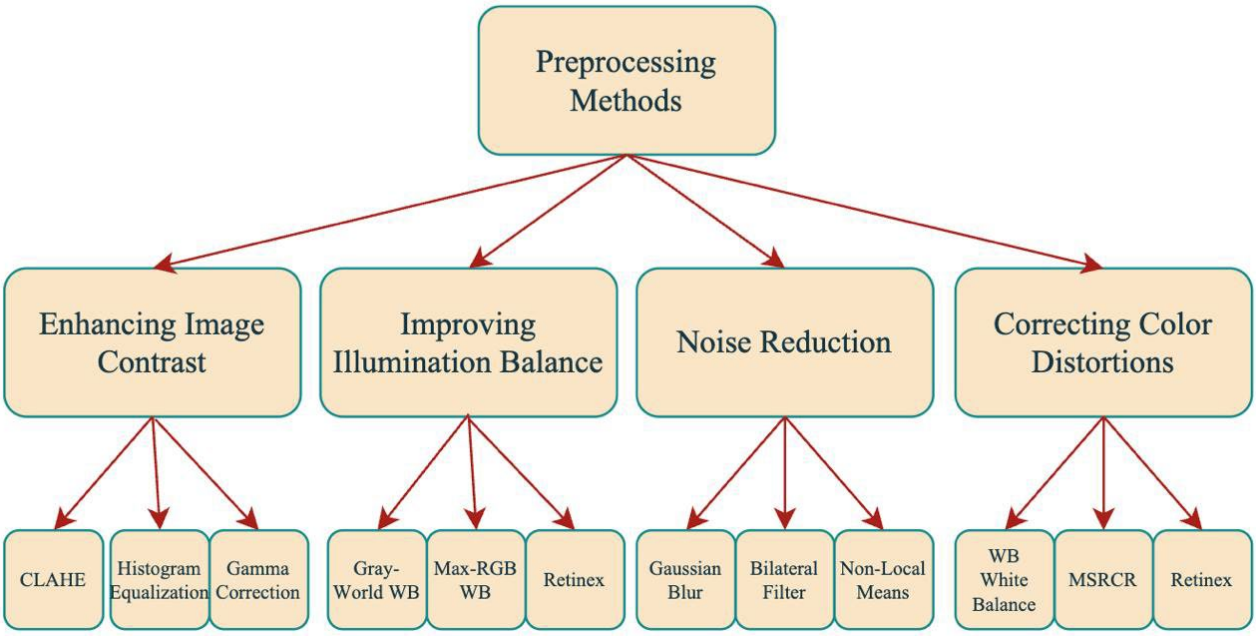

Figure 1 – Methods applied within the framework of the project.

In images captured under low-light conditions, issues such as low contrast, high noise levels, and uneven light distribution are commonly encountered, making accurate object detection challenging. The processing of such images has been extensively studied using a variety of preprocessing techniques, as illustrated in Figure 1. For instance, the CLAHE method enhances local contrast and improves image quality, while Gamma Correction adjusts brightness to restore a more natural appearance. Histogram Equalization balances bright and shadowed areas, and Gaussian Blur along with Bilateral Filtering reduces noise while preserving sharp edges. The Non-Local Means method performs filtering based on the similarity of image patches. Gray-World and Max-RGB techniques adjust color balance, while Retinex and MSRCR optimize lighting and color to provide natural saturation and contrast levels that approximate human visual perception. All of these methods have been implemented using the OpenCV library in the Python environment [18]. Each contributes to improved image quality and, when automatically optimized using genetic algorithms, significantly enhances the input data quality for the YOLOv5 model.

Convolutional Neural Networks (CNNs) are widely used for the automatic detection and localization of objects and play a crucial role in image processing. Deep neural networks, particularly CNN architectures, have broad applications in image analysis and object detection. A detailed explanation of their fundamentals and effectiveness is provided in [16]. The YOLO model represents one of the state-of-the-art techniques in this field. While earlier models such as YOLOv3 [9] have been successfully applied for object detection, YOLOv5–being a more advanced version–performs more effectively under low-light conditions. As shown in Figure 2, the YOLOv5 architecture processes the entire image area in a single pass, allowing real-time object detection with high accuracy [1]. Its architecture includes components such as an anchor-based head, a Feature Pyramid Network (FPN), and Cross Stage Partial (CSP) blocks, all of which contribute to its increased speed and performance.

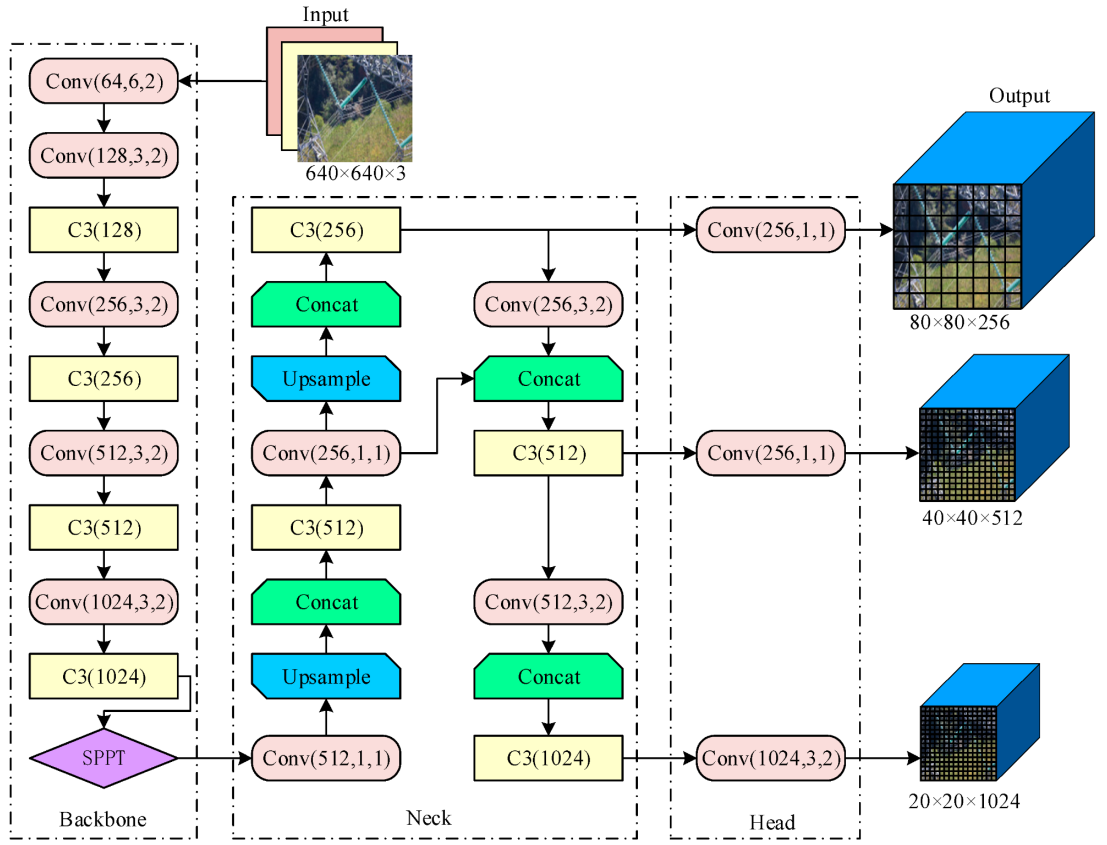

Figure 2 – YOLOv5 Architecture.

Manual selection of preprocessing method parameters is a labor-intensive and subjective process. To address this issue, genetic algorithms (GAs) were employed in this project. Genetic algorithms are optimization techniques inspired by the principles of natural evolution. While Particle Swarm Optimization (PSO) [11] is also widely studied in the field of evolutionary computation, the GA method was selected for this work due to its demonstrated effectiveness in filter parameter optimization. In this method, optimal parameters are automatically selected through the processes of population generation, selection, crossover, and mutation [12],[13].

DEAP (Distributed Evolutionary Algorithms in Python) is a library designed for the convenient implementation of genetic algorithms in Python [6]. The integration of DEAP with PyTorch enabled the development of high-performance optimization models [17]. In the project, each individual is represented as a 10-dimensional vector encoding a set of filter parameters. The fitness function is defined as the average confidence score obtained by the YOLOv5 model during object detection. To facilitate parallel computing, the Python multiprocessing module is used, enabling the concurrent evaluation of multiple individuals. This significantly reduces the overall training and optimization time.

The algorithmic structure of the project is organized according to the flowchart presented in Figure 3.

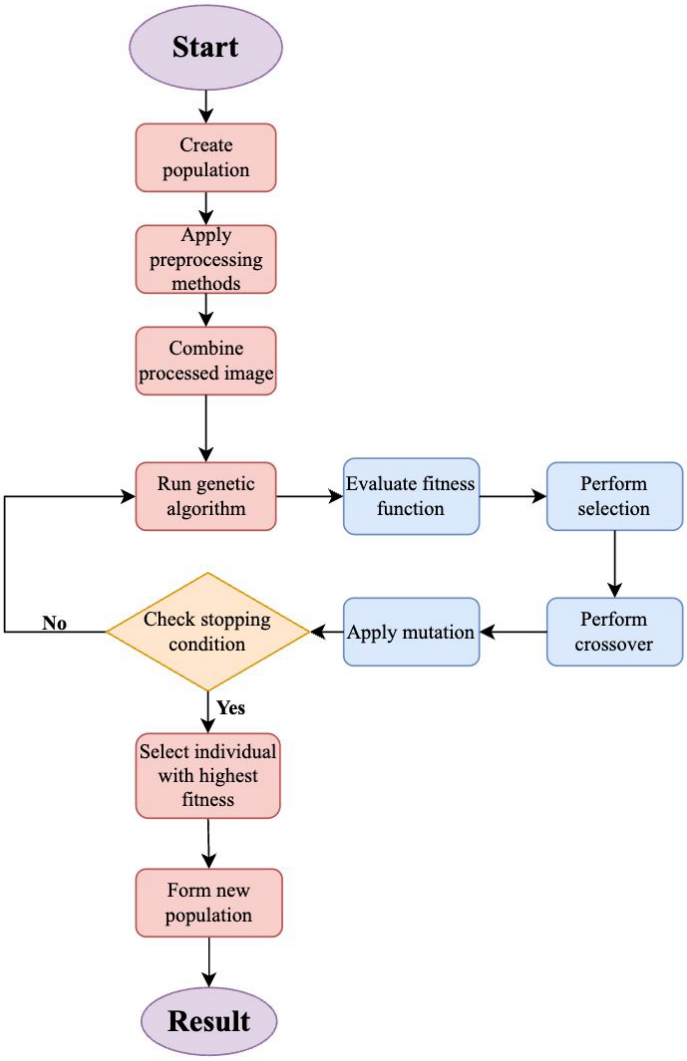

Figure 3 – Flowchart of the genetic algorithm for processing images captured under low-light conditions and optimizing the YOLOv5 model.

The operation of the genetic algorithm, as illustrated in Figure 3, begins with the formation of an initial random population. This population consists of multiple individuals, each representing a set of parameters used for the preprocessing of a specific image. Each individual is described by a parameter vector of the following form:

$$individual = [p_1, p_2, \ldots, p_{10}] \quad (1)$$

where: p_i – the corresponding preprocessing parameters applied in filters such as CLAHE, Gamma Correction, Histogram Equalization, Gaussian Blur, Bilateral Filter, Non-Local Means, Gray-World WB, Max-RGB WB, Retinex, and MSRCR.

Each of these filters is used to enhance image contrast, reduce noise, and improve lighting balance [2], [10], [5]. The parameter vectors allow for effective control over the image enhancement process. However, manual parameter selection often does not lead to optimal results, which creates a need for automated approaches. Studies have shown that evolutionary algorithms, including genetic algorithms, are highly effective in the automatic optimization of parameters [22].

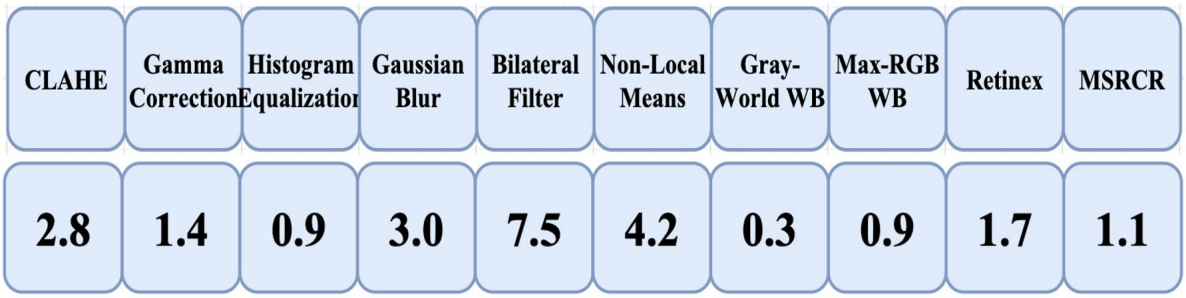

Figure 4 – Placement of preprocessing filter arrays.

These parameters are selected to enhance the video quality and are then evaluated using a fitness function. As shown in Figure 4, the preprocessing filters perform efficient vector operations using NumPy arrays, which increases computational performance [19].

Each individual (1) is used to process the image, and the resulting output is evaluated using the YOLOv5 model. Since the YOLOv5 model is widely studied in the literature as a real-time algorithm with high accuracy [1], [8], YOLOv5 detects objects in the processed image, and their confidence score determines the individual's effectiveness. This process is described by the following fitness function:

$$F(I_i) = \frac{1}{N_i} \sum_{i=1}^{N_i} C_j \quad (2)$$

where:

$F(I_i)$ – the fitness evaluation function of the YOLOv5 model for image I_i;

I_i – the number of detected objects;

N_i – the number of identified objects;

C_j – the confidence score for each detected object.

This function demonstrates how well the YOLOv5 model is able to confidently detect objects in the processed image. If the image is processed well, YOLOv5 will detect the objects with a high confidence score, resulting in a higher fitness value.

For example, let's assume that this individual, as shown in Figure 4, has the following parameters:

$$individual = \\ = [2.8, 1.4, 0.9, 3.0, 7.5, 4.2, 0.3, 0.9, 1.7, 1.1] \quad (3)$$

After passing the appropriately processed image to the YOLOv5 model, the following objects were detected. The YOLOv5 model detected the object "knife" with a confidence of 88%, "gun" with 76%, and "phone" with 53%.

$N_i = 3$ – number of detected objects

$$C_1 = 0.88,\ C_2 = 0.76,\ C_3 = 0.53$$

$$F(I_i) = \frac{1}{3}(0.88 + 0.76 + 0.53) = \\ = \frac{2.17}{3} = 0.7233$$

Thus, the fitness value of this individual is 0.723, which is a relatively good result. This function quantitatively evaluates the quality of the processed image based on the confidence values obtained from the YOLOv5 model. It serves as the basis for selecting individuals at the stages of selection, crossover, and mutation in the genetic algorithm, reflecting the effectiveness of preprocessing, which influences detection accuracy.

Selection is one of the important stages of the genetic algorithm. At this stage, individuals with the best fitness scores are chosen to be transferred to the next generation. In the project, tournament selection is used as the selection method. According to this method, several individuals are selected randomly, and the one with the highest fitness function value is chosen. Tournament selection is known for its simplicity and stability, making it popular in evolutionary problems [22]. This approach ensures the preservation of diversity in the population and prevents getting stuck in local minima.

After the selection phase, the genetic algorithm moves to the crossover phase, as shown in Figure 3. During the crossover stage, the parameters of the selected parent individuals are mixed, resulting in the formation of a new generation (offspring) of individuals.

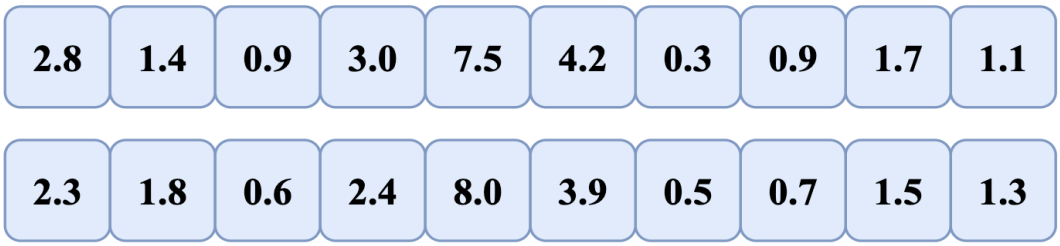

Figure 5 – First individual and second individual.

As shown in Figure 5, a descendant (offspring) is generated from two parent individuals using the BLX-α (Blend Crossover) method. This method was selected due to its effectiveness, and in this study, the α coefficient is set to 0.5. Each parameter of the new individual is calculated using the following formula:

$$C_k = [P_{min} - \alpha * d]\ U\ [P_{max} + \alpha * d] \quad (4)$$

where:

C_k – parameter of the offspring (child gene);

P_{max}, P_{min} – the maximum and minimum values of the corresponding parameter from the two parent individuals;

$\alpha \in [0,1]$ – blending coefficient (set to 0.5 in this study);

d = $P_{max} - P_{min}$ the difference between parent parameter values;

$U[.,.]$
– $uniform\ distribution\ over\ the\ interval$
$[a,b][a,b][a,b]$.

Example:

$$P_{max} = 2.8\ ,\ P_{min} = 2.3\ ,$$
$$\alpha = 0.5\ d = 2.8 - 2.3 = 0.5$$
$$C_1 = [2.3 - 0.5 * 0.5]\ U\ [2.8 + 0.5 * 0.5] =$$
$$= [2.05]\ U\ [3.05]$$

A value is randomly selected from this interval, for instance: $C_1 = 2.64$

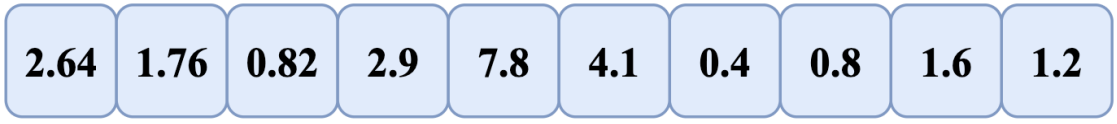

Figure 6 – Result obtained using the BLX crossover method.

Figure 6 presents a graphical interpretation of the BLX-α crossover method. This method extends the boundaries of the search space, thereby increasing the likelihood of the genetic algorithm reaching globally optimal solutions. Moreover, the BLX-α crossover reduces the risk of getting trapped in local extrema and enables a more thorough exploration of the solution space. These characteristics make it an effective and reliable approach for high-dimensional parameter optimization problems, such as in image preprocessin-g tasks [15].

Following the crossover stage, a mutation operation is applied to some of the individuals' parameters. This is a crucial mechanism for maintaining diversity within the population and for broadening the search space. The mutation process is based on the normal (Gaussian) distribution and is described by the following formula:

$$p_i' = p_i + N(0, \sigma^2) \quad (5)$$

where:

$p_i{}'$ – the parameter value after mutation;

p_i – the original parameter value;

$N(0, \sigma^2)$ – a normal distribution with zero mean and variance σ^2 .

In this study, $\sigma = 0.5$ which corresponds to a variance of 0.25. Thus, the random value is sampled from the distribution N(0,0.25).

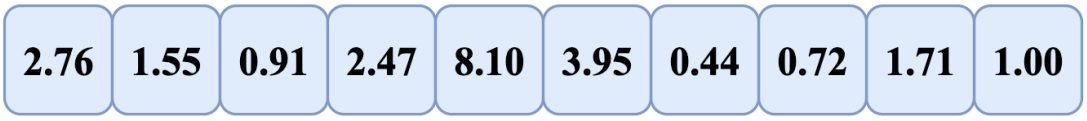

Figure 7 – Result after applying the mutation operator.

In this case, the mutation values are randomly drawn from the distribution N(0,0.25) and added to the original parameter values. This approach promotes the exploration of new local extrema of the fitness function and helps move closer to a globally optimal solution. Figure 7 shows the results after applying the mutation operation. It illustrates graphically that even small deviations in parameters allow for a deeper exploration of the solution space.

The DEAP (Distributed Evolutionary Algorithms in Python) library provides a simple and flexible framework for building genetic algorithms in Python. It includes pre-built components for population generation, fitness evaluation, selection, crossover, and mutation [6]. To ensure accurate object detection on low-light images using the YOLOv5 model, the integration of this architecture with a genetic algorithm demonstrated high precision even under challenging conditions [1].

One of the main limitations of genetic algorithms is the demand for substantial computational resources. This is particularly true during the fitness evaluation stage, as the YOLOv5 model must be applied to each individual, significantly increasing computation time. To address this issue, parallel computing methods were employed in the study. Using the Python multiprocessing module, the fitness function (2) was executed in parallel across multiple CPU cores. This approach allows for simultaneous evaluation of several individuals and significantly reduces the total computation time.

3. Results

This section presents the results of the experimental study conducted within the framework of the project. The experiments were based on images acquired from an X-ray scanner under low-light conditions, with the primary objective of detecting objects inside luggage.

The dataset used for this study is the X-ray Baggage Dataset, which contains 5,000 X-ray images and was sourced from Kaggle, a publicly available dataset repository. These images simulate real-world security screening conditions, including various concealed objects captured under low illumination. The dataset was divided into 80% for training and 20% for testing. All images were resized to a uniform resolution of 640×640 pixels, as required by the YOLOv5 model.

The dataset comprises 12 object classes, such as knife, gun, scissors, phone, battery, and others. The distribution of these classes is imbalanced, which was addressed using class weighting and data augmentation during training.

Experiments were conducted on a MacBook Pro with an 8-core Apple M1 processor and 6 GB unified memory, including 4 performance and 4 efficiency cores. Computations were executed using the Apple M1 GPU accelerated by Metal Performance Shaders (MPS). The software environment included Python 3.10, PyTorch (MPS backend), OpenCV, and Ultralytics YOLOv5. CUDA acceleration was not used.

The research was structured into the following stages: object detection on the original (unprocessed) X-ray images using the YOLOv5 model, evaluation of preprocessing techniques, measurement of YOLOv5 performance metrics (mAP, Precision, Recall), parameter optimization using genetic algorithms, and acceleration of the computational process using the multiprocessing module.

The study was organized into several key phases: object detection on the raw image, analysis of the impact of preprocessing techniques, evaluation of YOLOv5 performance, parameter optimization using genetic algorithms, and efficiency analysis of parallel computing methods.

At the initial stage, images captured under low-light conditions were fed into the YOLOv5 model without any preprocessing. Due to insufficient lighting, low contrast, and high noise levels, object detection accuracy was significantly reduced. In this case, the YOLOv5 model was only able to detect a subset of the objects. According to the experimental data, as shown in Figure 8, the average object detection confidence score on the original images was approximately 34%. This highlights the key

factors hindering the effective performance of YOLOv5 under low-light conditions [1],[10].

The annotated results of object detection on these images were used to visualize the exact object locations in the original state and to demonstrate the initial detection performance of the YOLOv5 model. These baseline results were subsequently compared with the outcomes obtained after applying the genetic algorithm and preprocessing techniques.

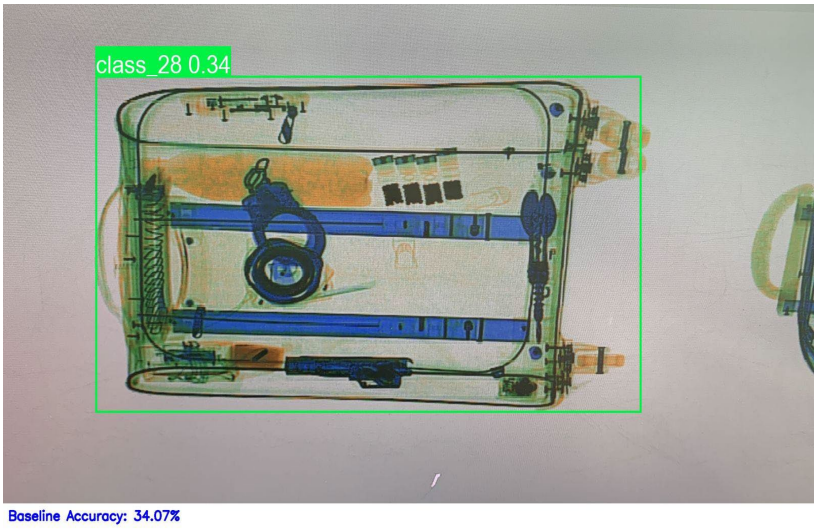

Figure 8. Object detection by the YOLOv5 model on the original image captured under low-light conditions.

Preprocessing methods are one of the crucial stages in image quality enhancement. In this project, techniques such as CLAHE, gamma correction, histogram equalization, Gaussian blurring, bilateral filtering, and methods like Retinex and MSRCR were employed [23],[24]. These methods help enhance local contrast, adjust lighting balance, and efficiently filter out noise. As a result, the quality of the data fed into the YOLOv5 model was significantly improved, directly impacting the object detection accuracy in low-light conditions. The conducted experiments demonstrated that the detection accuracy on preprocessed images was significantly higher than in the original state. In some cases, as shown in Figure 9, the YOLOv5 detection accuracy increased from 0% to 47%, which greatly enhanced the overall system performance.

Figure 9. Results of applying preprocessing methods.

As a result of the genetic algorithm application, the preprocessing parameters are automatically tuned, significantly improving the object detection accuracy by the YOLOv5 model. Figure 10 shows that at the 18th epoch, the detection accuracy reached 58.25%, representing a substantial improvement compared to the initial results. In this case, the following methods with parameters were used: CLAHE (1.05), Gamma Correction (1.77), Histogram Equalization (1.41), Gaussian Blur (6.98), Bilateral Filter (5.58), Non-Local Means (9.54), Gray-World WB (0.80), Max-RGB WB (0.35), Retinex (1.00), MSRCR (1.73).

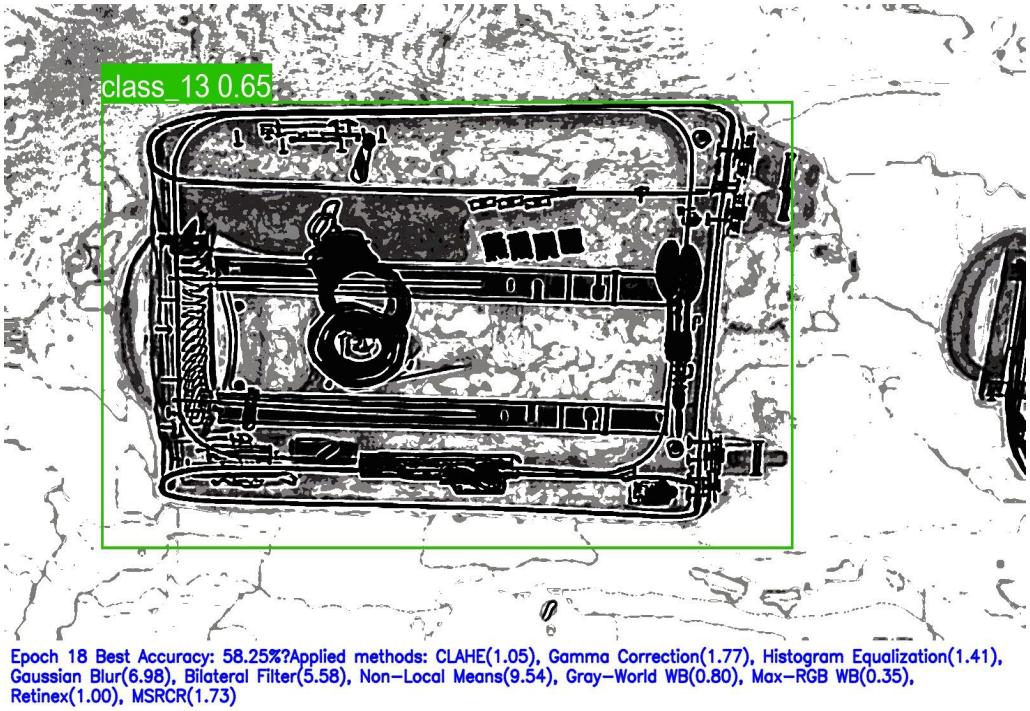

Figure 10. Best result of the 18th epoch achieved with the genetic algorithm and applied filters.

The genetic algorithm operated over 20 epochs, with images saved at each stage showing the best results. Each image displayed the detected objects, accuracy metrics, and applied filters. This visual comparison allows for the evaluation of the optimization process's effectiveness, as shown in Figure 11.

Object detection accuracy by the YOLOv5 model on the original image (left) and the image processed using the genetic algorithm (right) is shown in Figure 12. For the original image, the model's average confidence level was 34.07%, and after optimization, it increased to 82.81%. These results clearly demonstrate the effectiveness of the automatic filter parameter tuning and preprocessing, as genetic algorithms select the most effective image processing method, improving the quality of input data for the YOLOv5 model and increasing object detection accuracy.

In the project, genetic algorithms were implemented using the DEAP library (Distributed Evolutionary Algorithms in Python). This approach enabled the automatic optimization of filter parameters applied for image preprocessing [6]. During each epoch, the fitness values of individuals in the population gradually increased. According to the experimental results, as shown in Figure 13, the fitness value of the best individual, calculated based on the YOLOv5 model, reached a range of 85–87% by the final stage.

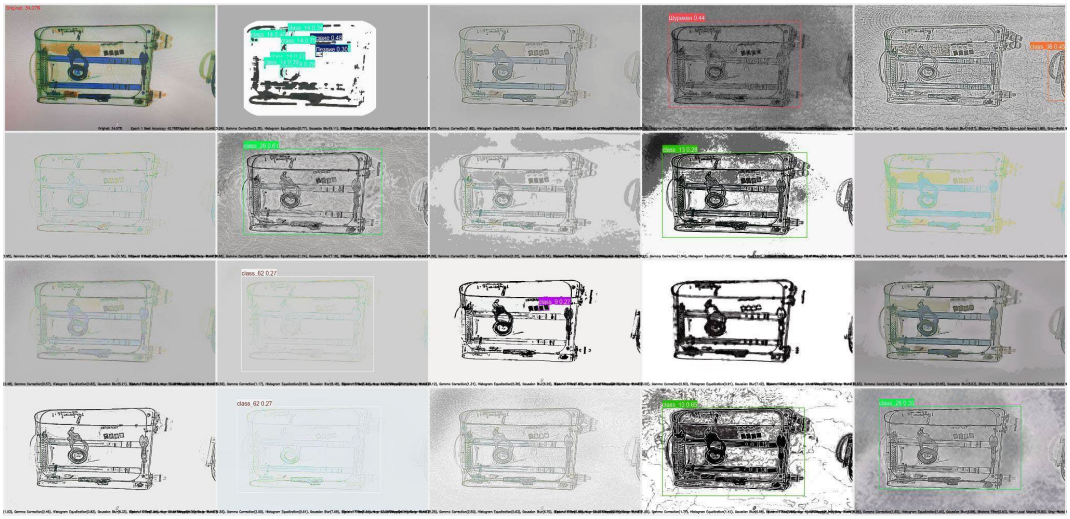

Figure 11. Collage composed of the best results from each epoch.

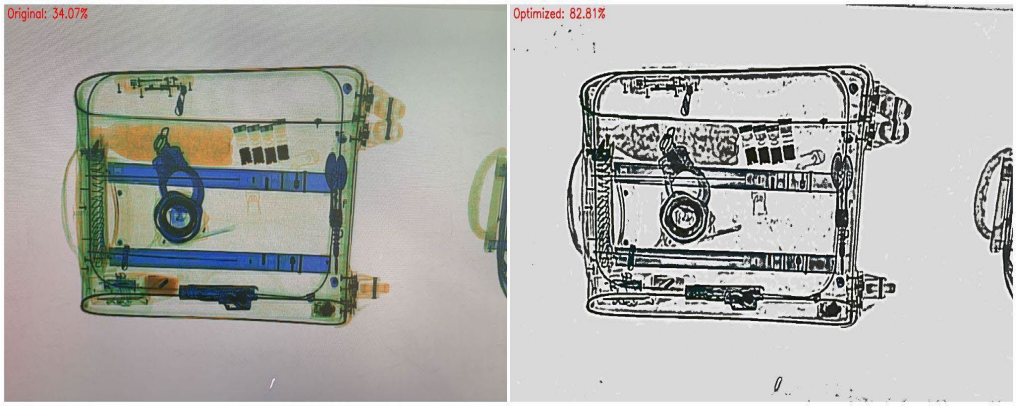

Figure 12. Original and processed images, along with their accuracy.

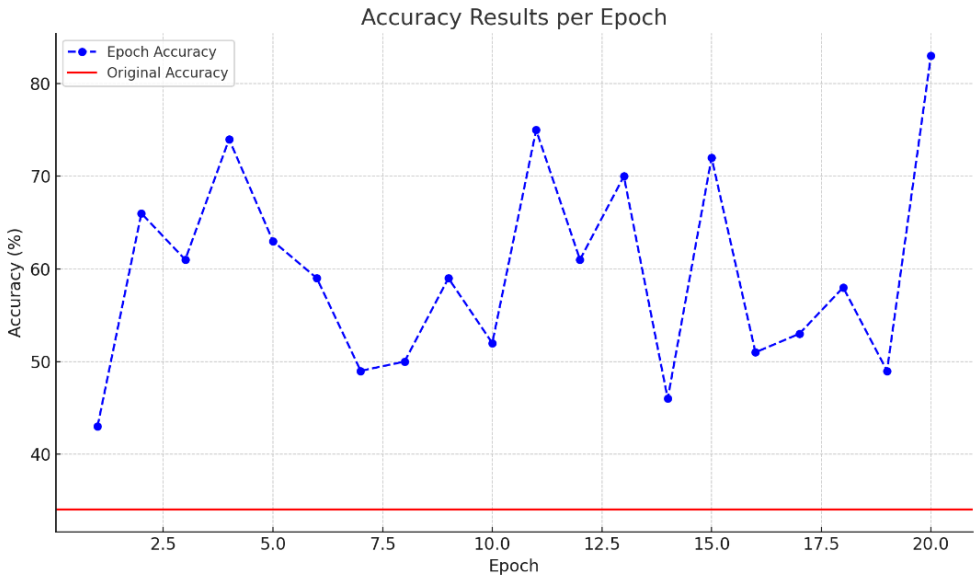

Figure 13. Accuracy chart across epochs.

Due to the high computational load of the genetic algorithm, parallel computing methods were used in the project to reduce overall processing time. By using the built-in Python module multiprocessing, the fitness value of each individual was calculated in parallel across multiple cores. This method significantly reduces computation time, especially when dealing with large populations.

Table 1. Comparison of accuracy and processing time on computers with 1 and 8 cores.

Population Size	Accuracy (1 core)	Time (1 core)	Accuracy (8 cores)	Time (8 cores)
30	81.21%	17 minutes	83.45%	5 minutes
50	86.55%	33 minutes	85.72%	9 minutes
100	87.71%	60 minutes	88.41%	16 minutes
200	83.92%	100 minutes	85.43%	30 minutes
Average	84.85%	52.5 minutes	85.75%	15 minutes

In the experiment, tests conducted on computers with 1 and 8 cores using the same parameters demonstrated that the use of 8-core parallel computing reduces the execution time by 80-90%. A more detailed comparison of the results is presented in Table 1 and Figure 14.

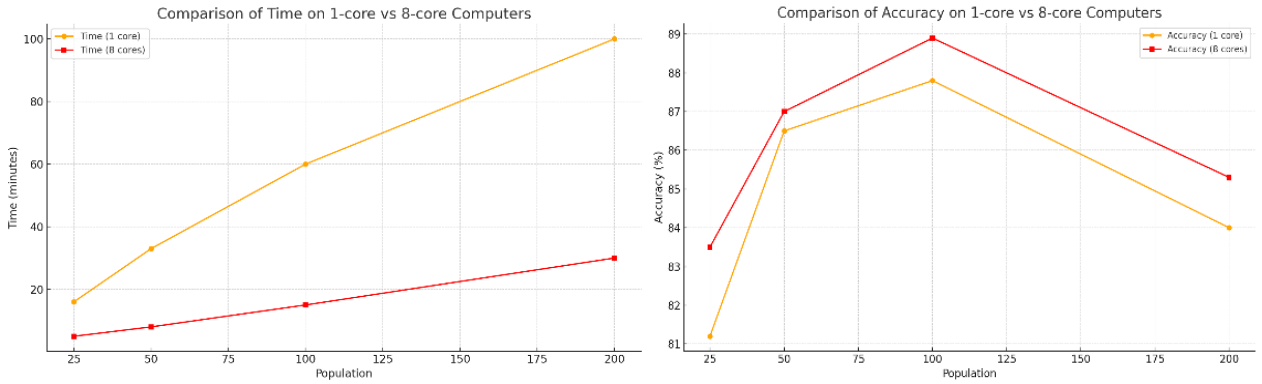

Figure 14. The relationship between accuracy and time versus population size (1 and 8 cores)

These results demonstrate that the combination of DEAP + YOLOv5 + Multiprocessing significantly enhances the system's time efficiency and makes it suitable for real-time object detection tasks. The experimental data presented show that when detecting objects in images captured under low-light conditions, higher accuracy and more efficient time performance are achieved compared to traditional methods [1], [2], [3], [6]. The obtained results confirm the practical applicability of the system for use in areas such as security monitoring, video surveillance, medical diagnostics, and autonomous control.

4. Conclusion

In conclusion, this study comprehensively addressed the issue of improving object detection accuracy on video data captured under low-light conditions and demonstrated the effectiveness of the integrated system. Modern preprocessing methods, such as CLAHE, gamma correction, histogram equalization, Gaussian blur, bilateral filter, as well as Retinex and MSRCR methods [2], [3], were applied to images obtained under low-light conditions. These methods enhanced image contrast, reduced noise, and significantly improved overall quality, which allowed for substantial improvements in the input data quality for the YOLOv5 model. The integration of preprocessing methods significantly improved YOLOv5's ability to detect objects in low-light conditions. Experimental data showed that the model's average confidence level increased significantly, indicating the practical applicability of the system for real-time use in fields such as security monitoring, video surveillance, and autonomous control [1].

During the study, genetic algorithms, implemented using the DEAP library, were used to automatically optimize filter parameters. The fitness values of the most effective individuals obtained through the GA steadily improved in each generation, which contributed to increased stability and accuracy of YOLOv5's results. Additionally, the application of parallel computing via the Python multiprocessing module allowed for a significant reduction in experimental execution time–on average, the time per epoch was reduced by 80–90%. This substantially improved the overall system performance and its efficiency for real-time use.

Thus, the proposed architecture, combining DEAP, YOLOv5, and parallel computing methods, has proven effective in improving object detection accuracy in images captured under low-light conditions and represents a promising solution for practical applications in demanding fields.

Overall, the proposed integrated system, combining the YOLOv5 model, preprocessing methods, and genetic algorithms, allowed for enhanced accuracy and reliability in detecting objects in images obtained under low-light conditions. This research demonstrated its practical applicability and made a significant contribution to the fields of security systems, monitoring, and autonomous control [1], [2], [6], [3]. Further research is recommended to deepen the study, improve parameter optimization methods, integrate additional object detection algorithms, and develop parallel computing architectures.

Funding

This research was funded by the science committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan grant number AP19579370.

Author Contributions

Author Contributions: Conceptualization, M.Zh. and Zh.B.; Methodology, M.Zh. and B.M.; Software, F.N.; Validation, M.Zh., F.N., and Zh.B.; Formal Analysis, M.Zh. and B.M.; Investigation, F.N. and M.Zh.; Resources, Zh.B. and B.M.; Data Curation, F.N.; Writing – Original Draft Preparation, F.N.; Writing – Review & Editing, M.Zh. and B.M.; Visualization, F.N.; Supervision, Zh.B. and B.M.; Project Administration, M.Zh..

Conflicts of Interest

The authors declare no conflict of interest.

References

1. G. Jocher et al., "YOLOv5," GitHub repository, 2020. [Online]. Available: https://github.com/ultralytics/yolov5
2. K. Zuiderveld, "*Contrast Limited Adaptive Histogram Equalization (CLAHE*)," in Graphics Gems IV, San Diego, CA, USA: Academic Press, 1994, pp. 474–485, doi: 10.1016/B978-0-12-336156-1.50061-6.
3. Z. Rahman, D. J. Jobson, and G. A. Woodell, "Retinex processing for automatic image enhancement," *J. Electron. Imaging*, vol. 13, no. 1, pp. 100–110, Jan. 2004, doi: 10.1117/1.1636183.
4. S. M. Pizer et al., "Adaptive histogram equalization and its variations," *Comput. Vis. Graph. Image Process*., vol. 39, no. 3, pp. 355–368, Sep. 1987, doi: 10.1016/S0734-189X(87)80186-X.
5. R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2008. [Online]. Available: https://www.pearson.com/us/higher-education/program/Gonzalez-Digital-Image-Processing-3rd-Edition/PGM241219.html
6. F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary Algorithms Made Easy," J. Mach. Learn. Res., vol. 13, pp. 2171–2175, Jul. 2012. [Online]. Available: https://jmlr.org/papers/v13/fortin12a.html
7. W. McKinney, Python for Data Analysis, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2017. [Online]. Available: https://www.oreilly.com/library/view/python-for-data/9781491957660/
8. A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint*, arXiv:2004.10934, Apr. 2020.
9. J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint*, arXiv:1804.02767, Apr. 2018.
10. D. J. Jobson, Z. Rahman, and G. A. Woodell, "A multiscale retinex for bridging the gap between color images and the human observation of scenes," *IEEE Trans. Image Process*., vol. 6, no. 7, pp. 965–976, Jul. 1997, doi: 10.1109/83.597272.
11. J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. ICNN'95 – *Int. Conf. Neural Networks*, Perth, WA, Australia, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
12. J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1975. [Online]. Available: https://mitpress.mit.edu/9780262581110/adaptation-in-natural-and-artificial-systems/
13. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA, USA: Addison-Wesley, 1989. [Online]. Available: https://dl.acm.org/doi/book/10.5555/534133

14. M. Srinivas and L. M. Patnaik, “Genetic algorithms: A survey,” *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994, doi: 10.1109/2.294849.

15. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput*., vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

16. F. Chollet, Deep Learning with Python. Shelter Island, NY, USA: Manning Publ., 2017. [Online]. Available: https://www.manning.com/books/deep-learning-with-python

17. A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” in *Adv. Neural Inf. Process. Syst*., vol. 32, pp. 8026–8037, 2019. [Online]. Available: https://papers.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

18. G. Bradski, “The OpenCV Library,” *Dr. Dobb’s J. Softw. Tools*, 2000. [Online]. Available: https://opencv.org/

19. S. Van der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: A structure for efficient numerical computation,” *Comput. Sci. Eng*., vol. 13, no. 2, pp. 22–30, Mar. 2011, doi: 10.1109/MCSE.2011.37.

20. C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Proc. 6th Int. Conf. Comput. Vision (ICCV*), Bombay, India, 1998, pp. 839–846, doi: 10.1109/ICCV.1998.710815.

21. E. H. Land and J. J. McCann, “Lightness and Retinex theory,” *J. Opt. Soc. Am*., vol. 61, no. 1, pp. 1–11, Jan. 1971.

22. A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing. *Berlin, Heidelberg*: Springer-Verlag, 2003. [Online]. Available: https://doi.org/10.1007/978-3-662-05094-1

23. J. Han, “Image Enhancement Method for Low-light Scenes Based on Retinex Theory and GAN,” *IEEE Access*, vol. 8, pp. 42318–42330, 2020, doi: 10.1109/ACCESS.2020.2977079.

24. W. Wang, X. Wu, X. Yuan, and Z. Gao, “A Retinex-Based Low-Light Image Enhancement Method with Noise Suppression,” *IEEE Trans. Image Process*., vol. 30, pp. 6367–6380, 2021, doi: 10.1109/TIP.2021.3092735.

Information about authors

Mukhtar Zhassuzak is a PhD student and lecturer at the Department of Computer Science, al-Farabi Kazakh National University (Almaty, Kazakhstan). His research interests include artificial intelligence and robotics. ORCID ID: 0000-0001-8164-8199.

Farida Narkeshova is a fourth-year undergraduate student at the Department of Computer Science, al-Farabi Kazakh National University (Almaty, Kazakhstan). Her academic focus includes computer science with a growing interest in artificial intelligence. ORCID ID: 0009-0005-9347-5578

Zholdas Buribayev is a PhD holder and lecturer at the Department of Computer Science, al-Farabi Kazakh National University (Almaty, Kazakhstan). His research is centered on artificial intelligence and robotics. ORCID ID: 0000-0002-3486-227X.

Bazargul Matkerim is a PhD in the Computer Science Department at Al-Farabi Kazakh National University (Almaty, Kazakhstan, bazargul.matkerim@gmail.com). Her research interests include parallel computing and applications of machine learning. ORCID ID: 0000-0002-5336-687X.

Submission received: 03 May, 2025.
Revised: 29 May, 2025.
Accepted: 30 May, 2025.