UDC 50.43.15 **DOI 10.52167/1609-1817-2025-136-1-246-259**

A.S. Bayegizova[1], A.M. Jumagaliyeva[2]✉, V.B. Rystygulova[2],
G.I. Mukhamedrakhimova[1], Zh. B. Lamasheva[1]
[1]L.N. Gumilyov Eurasian National University,
[2]K. Kulazhanov Kazakh University of Technology and Business, Astana, Kazakhstan,
E-mail: jumagalievaainur.m.@gmail.com

THE USE OF ARTIFICIAL INTELLIGENCE FOR OBJECT RECOGNITION BASED ON NEURAL NETWORKS

Abstract. This article is dedicated to the development and investigation of methods for automatic object recognition using artificial intelligence and convolutional neural networks. A standard dataset for object recognition, including images of various types, was used as the experimental base, allowing the creation of a model capable of effectively classifying objects under various conditions, such as changes in lighting, viewing angles, and image quality. The methodology includes data normalization and augmentation, stratified dataset splitting, and model architecture optimization to ensure a balance between computational efficiency and classification accuracy. The developed convolutional neural network features a layered architecture with activation functions and regularization methods, enabling the model to remain robust and adaptive. The results of the study demonstrated the high accuracy and reliability of the proposed model. The analysis of classification errors identified challenges related to the recognition of visually similar objects and proposed solutions to minimize them. This work emphasized the significance of using artificial intelligence and convolutional neural networks for automating object recognition, opening new perspectives for their application in intelligent systems and high-precision technologies.

Keywords. Artificial intelligence, neural networks, recognition, classification, images, normalization, augmentation, automation.

Introduction.

In the era of rapid advancements in artificial intelligence technologies, automatic object recognition has become an integral part of intelligent systems. In the transportation sector, the recognition of traffic signs plays a critical role in enhancing road safety and advancing autonomous vehicles. Automating this process reduces human error and contributes to decreasing the number of road traffic accidents. According to the World Health Organization, approximately 1.3 million people die annually due to RTAs, and implementing efficient traffic sign recognition systems can significantly reduce this figure.

The aim of this study is to develop and experimentally evaluate a convolutional neural network (CNN) model designed for the automatic classification of traffic signs. The German Traffic Sign Recognition Benchmark (GTSRB) dataset, consisting of more than 50,000 images across 43 classes of traffic signs, was utilized as the foundation for this research. The outcome of the study is the creation of a model capable of classifying traffic sign images with high accuracy and robustness against varying lighting conditions, viewing angles, and image quality.

The object of the study is the traffic sign images contained in the GTSRB dataset, while the subject is the processing and classification of these images using deep learning methods.

The relevance of the research lies in the need to improve road safety by automating the recognition of traffic signs and the growing demand for intelligent transportation systems. Despite the significant progress in AI technologies, traffic sign recognition remains challenging due to the diverse conditions under which data is collected. The novelty of this research lies in

the development of a compact and highly efficient CNN architecture capable of addressing classification challenges even under constrained data and computational resources.

This study also addresses the existing challenges in automatic traffic sign recognition, including variations in lighting, angles, and image quality. Proposed solutions include preprocessing techniques such as normalization and data augmentation, as well as the optimization of neural network architecture and model hyperparameters.

Literature Review.

The task of traffic sign recognition has garnered significant attention in recent years due to its critical role in intelligent transportation systems and autonomous vehicles. Traditional methods for traffic sign recognition relied heavily on handcrafted features, such as histogram of oriented gradients and scale-invariant feature transform. While these methods demonstrated reasonable performance, their dependence on feature engineering limited their adaptability to diverse datasets and varying conditions, such as changes in lighting or occlusion.

The advent of deep learning, particularly Convolutional Neural Networks, has revolutionized the field by enabling automatic feature extraction from raw images. Modern architectures, such as ResNet, Inception, and MobileNet, have been widely adopted for traffic sign recognition tasks. ResNet, with its residual connections, allows for deeper networks by addressing the vanishing gradient problem, leading to improved accuracy according to recent research [1]. Inception models employ parallel convolutional layers of varying kernel sizes, enabling multi-scale feature extraction. MobileNet, on the other hand, focuses on computational efficiency, making it suitable for deployment on resource-constrained devices. Despite their success, these architectures often come with trade-offs between accuracy and computational cost [2].

Recent studies [3]-[5] explored the YOLO (You Only Look Once) models, widely used for real-time object detection, have also been applied to traffic sign recognition. YOLOv4, in particular, achieves impressive accuracy by combining novel components such as CSPDarknet53 and path aggregation networks. However, YOLO models are more computationally intensive, making them less suitable for lightweight embedded systems.

The German Traffic Sign Recognition Benchmark (GTSRB) dataset has emerged as the standard for evaluating traffic sign recognition methods. Studies utilizing this dataset have shown that class imbalance and visual similarity between signs can significantly affect model performance [6]. Techniques such as data augmentation, transfer learning, and ensemble methods have been proposed to address these challenges. However, the need for a compact yet high-performing model for real-time applications remains largely unmet as it underscored in [7]-[8].

This study builds on existing research by proposing a lightweight CNN architecture that balances high accuracy with computational efficiency. Unlike traditional approaches or larger architectures like ResNet and YOLO, the proposed model is optimized for real-time inference on devices with limited resources. Furthermore, it incorporates stratified dataset splitting and data augmentation to address class imbalance and ensure robust generalization across diverse traffic sign categories.

Materials and Methods.

This study developed an object recognition model based on neural networks, focusing on the use of Convolutional Neural Networks for image processing and classification.

Data Collection and Preparation

The model was trained using the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which includes over 50,000 traffic sign images distributed across 43 classes. Each class represents a specific type of traffic sign, making the dataset highly representative of real-world road scenarios.

The images underwent preprocessing to standardize their dimensions to 32×32 pixels, and pixel values were normalized to the range [0, 1] to ensure the stable performance of the model. Data augmentation techniques, such as image rotation, shifting, and scaling, were applied to increase the volume of training data. These methods enhanced the model's ability to generalize and reduced the risk of overfitting.

Let us visualize all 43 class types using the coloumn `ClassId`

Ввод [5]:
```
num_classes = len(df_meta.ClassId.unique())
class_dict = {}
class_labels = list(range(num_classes))
# Speed Class 0-9
speed_class = ['Speed Limit ' + item for item in [speed + ' kmph' for speed in ['20', '30', '50', '60', '70', '80']]]\
            + ['End of Speed Limit 80 kmph']
speed_class+= ['Speed Limit ' + item for item in [speed + ' kmph' for speed in ['100', '120']]]
speed_class
# 10, 11 No Passing
no_pass = ['No Passing' + item for item in ['', ' vehicle over 3.5 ton']]
# 12-43
rest = ['Right-of-way at intersection', 'Priority road', 'Yield', 'Stop', 'No vehicles', 'Veh > 3.5 tons prohibited',\
          'No entry', 'General caution', 'Dangerous curve left', 'Dangerous curve right', 'Double curve', 'Bumpy road',
          'Slippery road', 'Road narrows on the right', 'Road work', 'Traffic signals', 'Pedestrians', 'Children crossing',
          'Bicycles crossing', 'Beware of ice/snow','Wild animals crossing', 'End speed + passing limits', 'Turn right ahead',
          'Turn left ahead', 'Ahead only', 'Go straight or right', 'Go straight or left', 'Keep right', 'Keep left',
          'Roundabout mandatory', 'End of no passing', 'End no passing vehicle > 3.5 tons']
class_values = speed_class + no_pass + rest
class_dict = {keys:values for keys,values in zip(class_labels, class_values)}
```

Ввод [6]:
```
sortFunction = lambda x: int(os.path.basename(x)[:-4])
plt.figure(figsize = (25, 25))
for i, imagename in enumerate(sorted(glob.glob(data_path + 'Meta/' + '*.*'), key = sortFunction)):
    plt.subplot(7, 7, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    plt.xlabel(class_dict[i])
    image = cv2.imread(imagename)
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.show()
```

Figure 1 - Data preparation stage

This figure 1 shows the categorical distribution of traffic signs into 43 classes, categorized based on their characteristics, such as speed limits, warning signs, and prohibitions. Each sign is visually represented to provide a comprehensive view of the dataset, emphasizing the diversity and balance needed for effective classification (Figure 1).

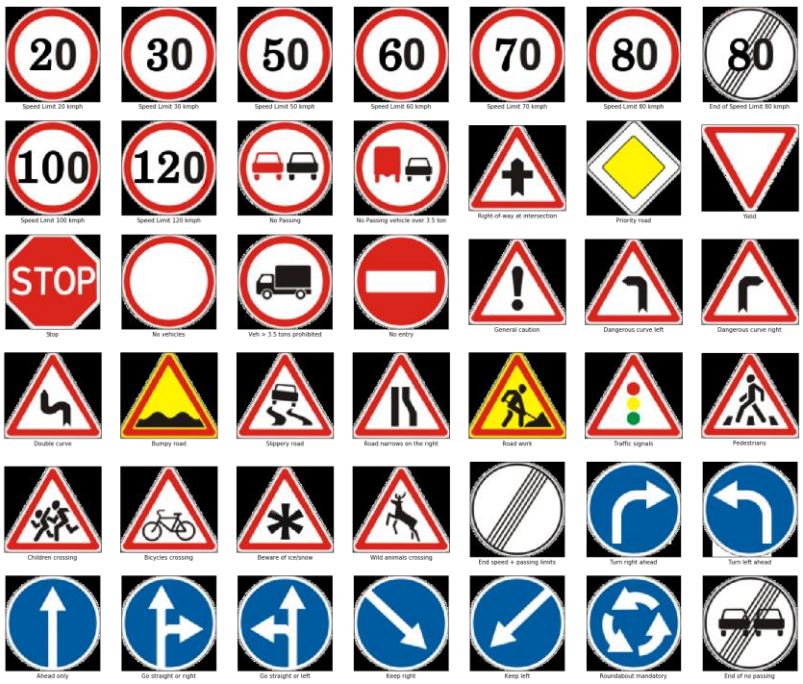

Figure 2 - Traffic sign classes for further work

As Figure 2 shows, the data preparation stage involved image normalization, stratified dataset splitting, and verification. Pixel values were normalized to the range [0, 1] by dividing by 255, improving gradient descent convergence and accelerating training (Figure 2). The dataset was stratified to maintain class proportions, with 80% allocated for training and 20% for validation. The final distribution included 31,367 training images, 7,842 validation images, and 12,630 test images, ensuring balanced representation across all classes.

Summary of Parameters:

- Total Parameters: 356,939.
- Trainable Parameters: 356,939.
- Non-trainable Parameters: 0.

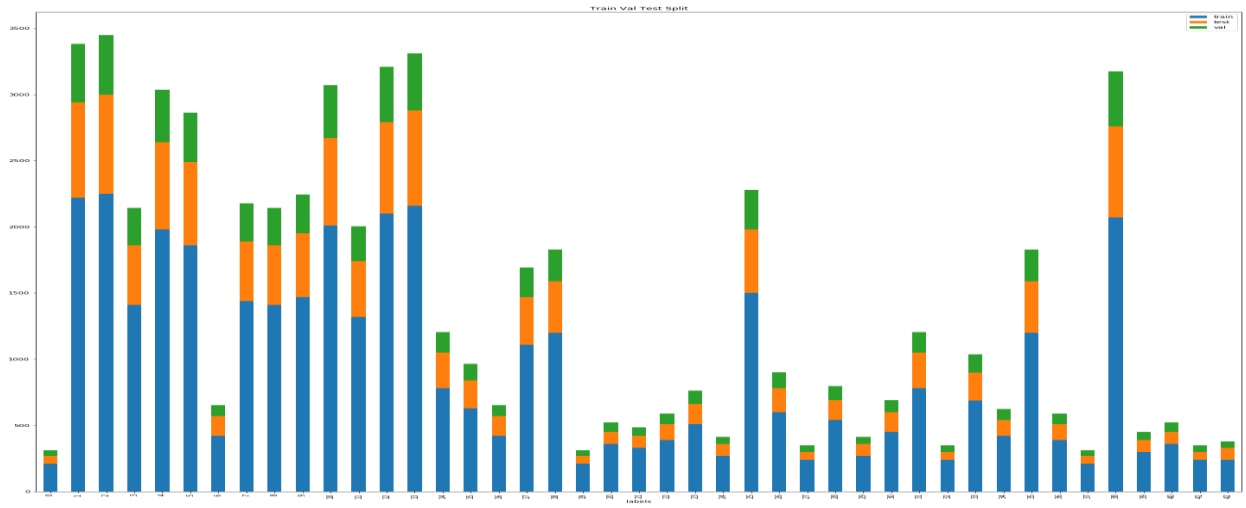

Figure 3 - Data distribution across subsets

The figure 3 illustrates the distribution of data across the training, validation, and test subsets for each of the 43 traffic sign classes. Maintaining proportional data distribution across all classes at every stage of the study was critical, as imbalanced data could skew results and degrade classification performance (Figure 3). The graph shows that each subset contains a proportional number of images for all classes, ensuring an objective evaluation of the model. This balance was achieved through the use of a stratified approach during data splitting. Equal representation of each class across subsets improved the model's generalization capability during training and testing.

Model Architecture. The architecture of the Convolutional Neural Network for traffic sign classification is detailed below. Each layer is described with its type, parameters, output dimensions, and the number of trainable parameters.

Table 1 - Convolutional Neural Network architecture

Layer	Type	Parameters	Output Dimensions	Number of Parameters
Input Layer	Input	(32, 32, 3) (image dimensions)	(32, 32, 3)	0
Convolutional Layer 1	Conv2D	32 filters, kernel (5×5), ReLU activation	(28, 28, 32)	2,432

Pooling Layer 1	MaxPooling2D	Kernel (2×2)	(14, 14, 32)	0
Dropout Layer 1	Dropout	Dropout rate: 0.2	(14, 14, 32)	0
Convolutional Layer 2	Conv2D	64 filters, kernel (3×3), ReLU activation	(12, 12, 64)	18,496
Pooling Layer 2	MaxPooling2D	Kernel (2×2)	(6, 6, 64)	0
Dropout Layer 2	Dropout	Dropout rate: 0.5	(6, 6, 64)	0
Fully Connected Layer 1	Flatten	-	(2,304)	0
Fully Connected Layer 2	Dense	256 neurons, ReLU activation	(256)	590,080
Dropout Layer 3	Dropout	Dropout rate: 0.5	(256)	0
Output Layer	Dense	43 neurons, Softmax activation	(43)	11,051

Table 1 illustrates the architecture of the convolutional neural network (CNN) developed for traffic sign classification. The architecture was designed with the complexity of the task in mind, as well as the requirement to process images of 32×32 pixels. The use of two convolutional blocks with ReLU activation functions and MaxPooling layers enables the model to effectively extract spatial features while reducing the dimensionality of the data [9].

The inclusion of Dropout layers with rates of 0.2 and 0.5 reduced the risk of overfitting, ensuring stable training. A fully connected layer with 256 neurons and ReLU activation processes the extracted features, while the output layer with a Softmax activation function classifies the images into 43 classes. The model's architecture is balanced in terms of parameter count and complexity, making it suitable for real-world applications (Table 1). This structure was selected after analyzing various approaches, and its effectiveness has been validated experimentally.

```
: history = model.fit(X_train, y_train, batch_size=32, epochs=20, validation_data=(X_val, y_val), verbose = True)

Train on 31367 samples, validate on 7842 samples
Epoch 1/20
31367/31367 [==============================] - 24s 759us/step - loss: 1.2779 - acc: 0.6313 - val_loss: 0.1863 - val_acc: 0.9517
Epoch 2/20
31367/31367 [==============================] - 13s 401us/step - loss: 0.2477 - acc: 0.9254 - val_loss: 0.1028 - val_acc: 0.9670
Epoch 3/20
31367/31367 [==============================] - 13s 417us/step - loss: 0.1548 - acc: 0.9532 - val_loss: 0.0624 - val_acc: 0.9841
Epoch 4/20
31367/31367 [==============================] - 14s 433us/step - loss: 0.1217 - acc: 0.9624 - val_loss: 0.0423 - val_acc: 0.9888
Epoch 5/20
31367/31367 [==============================] - 12s 387us/step - loss: 0.0951 - acc: 0.9702 - val_loss: 0.0385 - val_acc: 0.9898
Epoch 6/20
31367/31367 [==============================] - 14s 435us/step - loss: 0.0845 - acc: 0.9746 - val_loss: 0.0324 - val_acc: 0.9911
Epoch 7/20
31367/31367 [==============================] - 13s 424us/step - loss: 0.0771 - acc: 0.9763 - val_loss: 0.0330 - val_acc: 0.9921
Epoch 8/20
31367/31367 [==============================] - 14s 436us/step - loss: 0.0721 - acc: 0.9788 - val_loss: 0.0282 - val_acc: 0.9925
Epoch 9/20
31367/31367 [==============================] - 14s 432us/step - loss: 0.0650 - acc: 0.9802 - val_loss: 0.0285 - val_acc: 0.9941
Epoch 10/20
31367/31367 [==============================] - 13s 417us/step - loss: 0.0636 - acc: 0.9813 - val_loss: 0.0238 - val_acc: 0.9952
Epoch 11/20
31367/31367 [==============================] - 13s 410us/step - loss: 0.0623 - acc: 0.9819 - val_loss: 0.0235 - val_acc: 0.9945
Epoch 12/20
31367/31367 [==============================] - 12s 378us/step - loss: 0.0570 - acc: 0.9829 - val_loss: 0.0225 - val_acc: 0.9948
Epoch 13/20
31367/31367 [==============================] - 12s 381us/step - loss: 0.0572 - acc: 0.9836 - val_loss: 0.0247 - val_acc: 0.9938
Epoch 14/20
31367/31367 [==============================] - 14s 443us/step - loss: 0.0498 - acc: 0.9856 - val_loss: 0.0192 - val_acc: 0.9960
Epoch 15/20
31367/31367 [==============================] - 12s 394us/step - loss: 0.0496 - acc: 0.9860 - val_loss: 0.0258 - val_acc: 0.9946
Epoch 16/20
31367/31367 [==============================] - 12s 392us/step - loss: 0.0520 - acc: 0.9844 - val_loss: 0.0260 - val_acc: 0.9946
Epoch 17/20
31367/31367 [==============================] - 11s 365us/step - loss: 0.0525 - acc: 0.9857 - val_loss: 0.0223 - val_acc: 0.9955
Epoch 18/20
31367/31367 [==============================] - 11s 358us/step - loss: 0.0471 - acc: 0.9862 - val_loss: 0.0191 - val_acc: 0.9955
Epoch 19/20
31367/31367 [==============================] - 11s 357us/step - loss: 0.0478 - acc: 0.9868 - val_loss: 0.0265 - val_acc: 0.9952
Epoch 20/20
31367/31367 [==============================] - 11s 358us/step - loss: 0.0467 - acc: 0.9869 - val_loss: 0.0230 - val_acc: 0.9944
```

Figure 4 - Trained AI model

Figure 4 shows, that the training of the Convolutional Neural Network (CNN) was conducted using a batch size of 32 over 20 epochs, with validation performed on a separate subset (X_val and y_val). The training dataset consisted of 31,367 images, and metrics such as loss and accuracy were monitored at each epoch for both the training and validation datasets. The model exhibited a steady reduction in loss values and consistent improvement in accuracy, achieving a final training accuracy of 98.69% and a validation accuracy of 99.44%, indicating effective generalization without overfitting (Figure 4). The results, as illustrated in Figure 15, demonstrate the model's stability and readiness for further evaluation on the test dataset to validate its performance under real-world conditions.

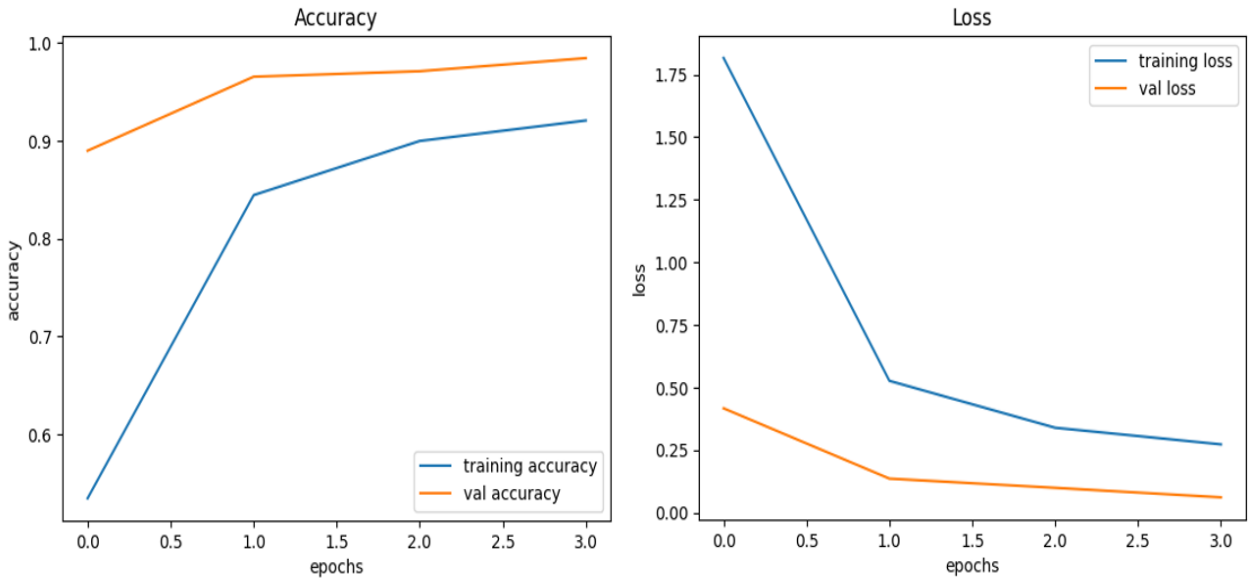

Figure 5 - Accuracy and Loss Plots

As the Figure 5 shows, the training process was analyzed using accuracy and loss plots as shown in Figure 5, which illustrate the model's performance on both training and validation datasets across epochs. The accuracy plot shows a consistent increase in accuracy, with training accuracy reaching 98.69% and validation accuracy stabilizing at ~99.44% by the 20th epoch. The minimal gap between the two curves indicates an absence of overfitting. The loss plot demonstrates a steady decline in loss values, with training loss reducing to approximately 0.05 and validation loss stabilizing at 0.02, highlighting the efficiency of the model's training process. These results confirm the model's stability, high accuracy, and effective learning, as evidenced by the synchronized trends in both metrics across datasets.

Testing the Model. After successful training, the model was tested on an independent dataset comprising 12,630 images to evaluate its performance in real-world conditions. To analyze classification accuracy, a confusion matrix was generated, as shown in Figure 6.

The confusion matrix indicates that most images were classified correctly, as most values are concentrated along the diagonal. However, the model made errors in certain cases. For example, visually similar traffic signs, such as speed limits of 30 km/h and 50 km/h, were sometimes misclassified. These errors are attributed to the high degree of similarity in shapes and symbols between these signs, which presents challenges for classification.

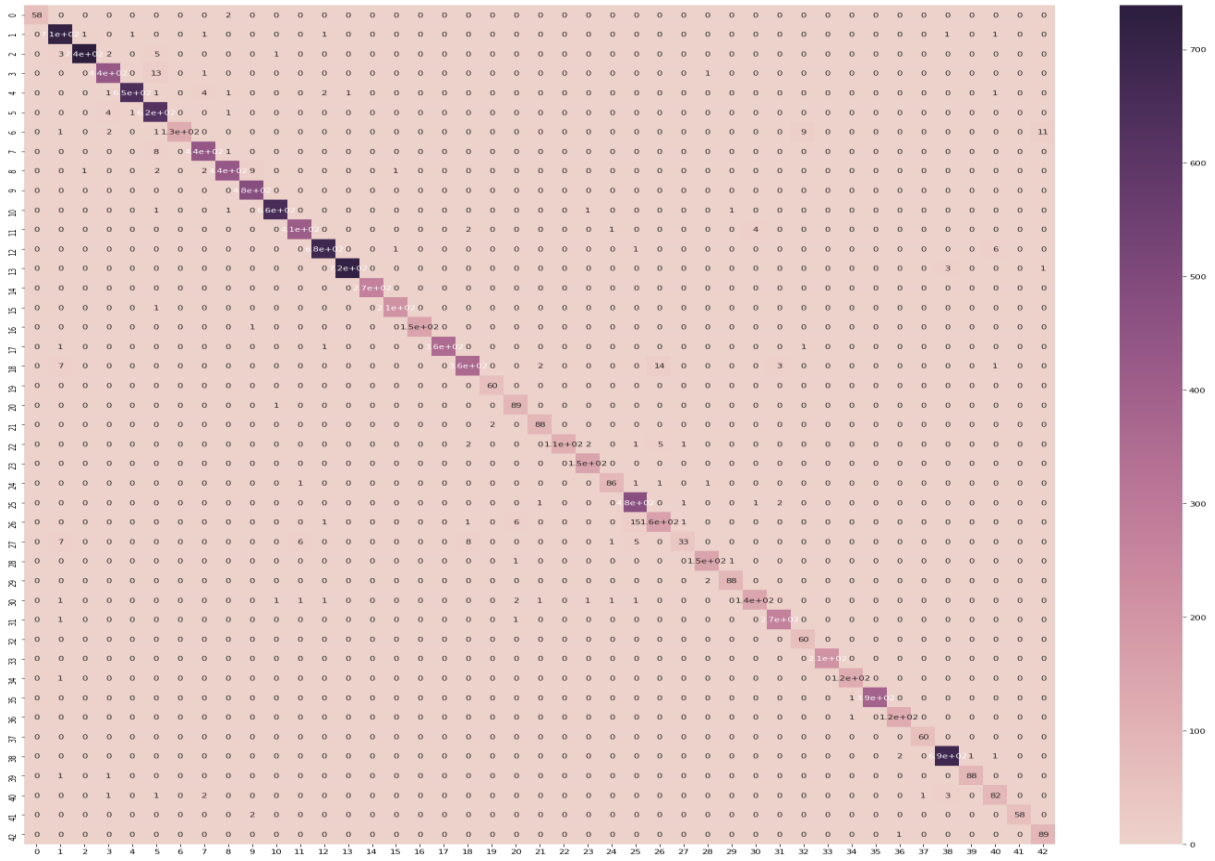

Figure 6 - Testing the model and Confusion Matrix

The confusion matrix shown in Figure 6 is one of the key tools for analyzing the model's performance. It provides a detailed view of which classes were classified correctly and which posed challenges. Most of the values are concentrated along the diagonal, indicating high overall accuracy. However, minor errors are observed between visually similar classes, such as speed limits of 30 km/h and 50 km/h. These errors can be attributed to the data's characteristics, such as minimal visual differences between these signs.

The confusion matrix provides valuable insights into potential areas for improving the model:

1. Expanding the Dataset: Adding more examples for classes with higher misclassification rates can improve the model’s robustness.
2. Reinforcement Learning: Leveraging reinforcement learning techniques can help the model focus on challenging classes and improve its decision-making process.
3. Enhanced Data Augmentation: Applying additional augmentation techniques can increase the variability of training data, highlighting subtle differences between visually similar classes.

Prediction Visualization. To qualitatively evaluate the model’s performance, prediction visualization was conducted, as shown in Figure 7, Figure 8 and Figure 9. Each image from the test set is paired with its actual and predicted class. Correct predictions are marked with green labels, while errors are highlighted with red labels.

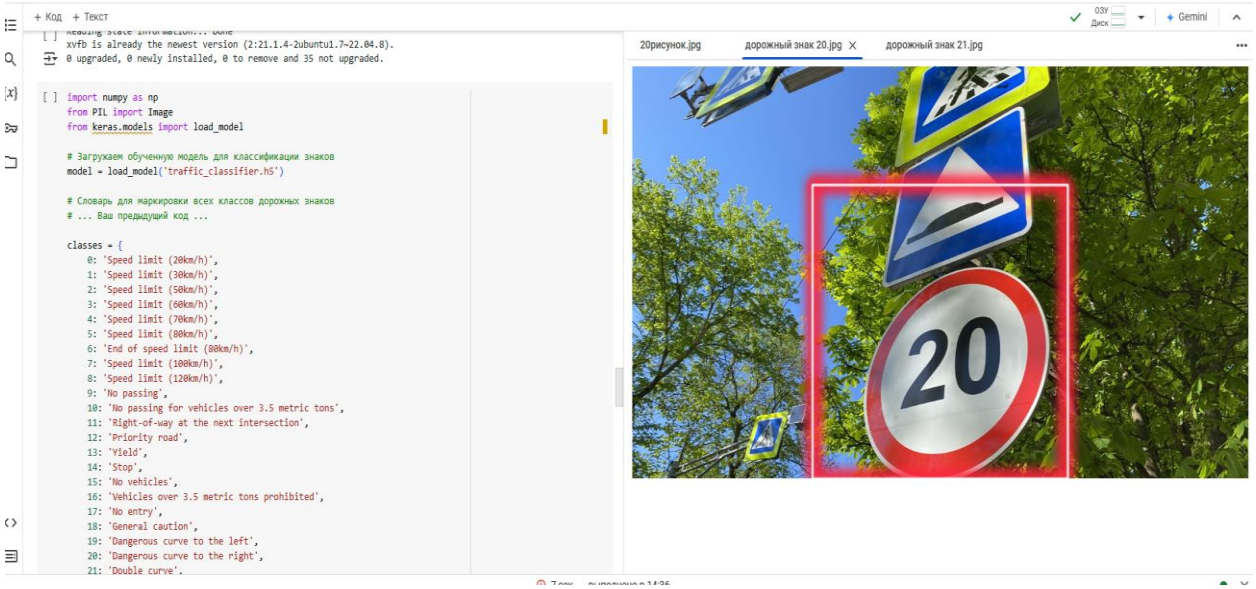

Figure 7 - Example of Model Classification on Individual Traffic Signs

This figure illustrates the model's ability to classify individual traffic signs accurately. It highlights the input image, the predicted class, and the model's detection capabilities. This visualization demonstrates the effectiveness of the trained model in identifying specific traffic signs in real-world scenarios. Such examples are crucial for validating the model's performance in practical applications.

Figure 8 - Multi-Sign Detection and Classification in Complex Scenes

Figure 8 demonstrates the model's ability to detect and classify multiple traffic signs in real-world scenarios with overlapping signs, varying sizes, and challenging conditions like background clutter. This showcases the model's robustness and practical applicability in dynamic environments, such as autonomous driving, where accurate recognition of multiple signs is crucial for safety and navigation.

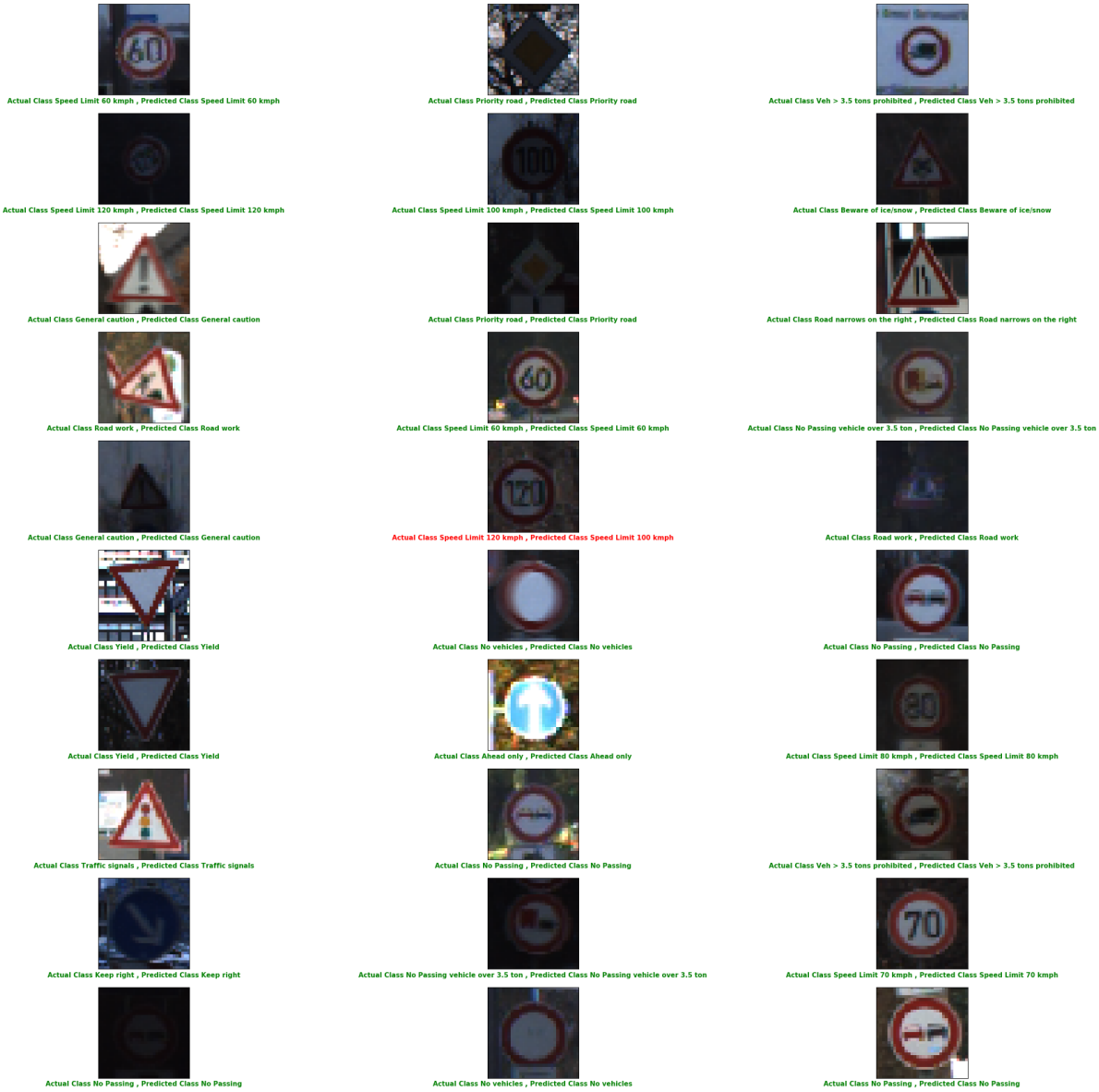

Figure 9 - Prediction visualization

Figure 9 provides a visualization of the model's predictions on random images from the test set. Each image is labeled with its actual class and the class predicted by the model. Correct predictions are highlighted in green, while misclassifications are marked in red. This visualization allows for an evaluation of the model's performance on individual examples and helps identify patterns in its errors (Figure 9). The green labels indicate correctly classified signs, while the red labels point to misclassified examples. The misclassifications often occur between visually similar signs, such as speed limits of 30 km/h and 50 km/h, reflecting the model's challenges in distinguishing minor visual differences. This analysis underscores the importance of refining the dataset and model architecture to minimize errors. Additionally, it demonstrates the effectiveness of the model in most scenarios, validating its potential for real-world traffic environments.

Results and Discussion.

Table 2 shows the developed convolutional neural network (CNN) for traffic sign recognition demonstrated outstanding performance, achieving a training accuracy of 98.69%, a

validation accuracy of 99.44%, and a test accuracy of 99.02%. The model effectively generalized across diverse traffic sign classes, showcasing its robustness in handling variations in lighting, angles, and image quality (Table 2).

Table 2- Summary of Model Performance Metrics

Category	Metric	Result	Commentary
Training	Accuracy on training data	98.69%	The model was trained effectively, demonstrating a steady increase in accuracy without signs of overfitting.
Validation	Accuracy on validation data	99.44%	High accuracy indicates the model's ability to generalize to unseen data.
Testing	Accuracy on test data	99.02%	The model's reliability was confirmed on an independent dataset of 12,630 images.
Confusion Matrix	Percentage of correct classifications	High	Most errors occur between visually similar classes (e.g., speed limits of 30 and 50).
Model Simplicity	Total number of parameters	356,939	The model is optimized for practical use on resource-constrained devices.
Classes with Highest Error	Similar traffic signs	Speed limits and related signs	Errors are linked to minimal visual differences between signs.
Overall Performance	Classification accuracy	High	The model demonstrated consistent performance across all stages of testing and training.

The confusion matrix (Figure 17) highlighted that most misclassifications occurred between visually similar classes, such as speed limits of 30 km/h and 50 km/h. These errors are likely due to minimal visual differences and an imbalanced dataset for certain classes. While the overall accuracy remained high, these findings underscore the need for enhanced preprocessing and architectural adjustments.

Also, the high accuracy results validate the use of CNNs for traffic sign classification, emphasizing their ability to extract meaningful features from image data. Key factors contributing to the model's success include balanced data distribution, normalization, and data augmentation techniques. These methods enhanced the model's ability to generalize across unseen data.

However, challenges remain, primarily related to visually similar traffic signs. Augmenting the dataset with additional samples for rare classes or employing more advanced architectures, such as ResNet or ensembles, could address these issues. Furthermore, applying transfer learning on larger datasets or real-world traffic sign collections could enhance the model’s performance in diverse conditions. The model’s compact architecture, with 356,939 parameters, makes it suitable for deployment in real-time applications, such as Advanced Driver Assistance Systems (ADAS). Its ability to operate on devices with limited computational resources positions it as a valuable tool for modern intelligent transportation systems [10].

Table 3 - Comparison of the Proposed Model with Other Architectures

Model	Test Accuracy (%)	Number of Parameters	Inference Time (per image)	Key Features
Proposed Model	99.02	356,939	~3.1 ms	Compact architecture suitable for embedded systems; high accuracy on balanced data.
YOLOv4	96.2	~65M	~25 ms	Fast object detection and classification; high computational requirements.
ResNet-50	98.1	~23.5M	~12 ms	Deep architecture with high accuracy but requires more data and computational resources.
MobileNetV2	97.3	~3.4M	~7 ms	Optimized for mobile devices; struggles with complex data.
InceptionV3	98.6	~22M	~18 ms	Complex architecture designed for classifying intricate images.

The results in Table 3 highlight the proposed model's efficiency, achieving 99.02% test accuracy with only 356,939 parameters, outperforming heavier models like YOLOv4 and ResNet-50 in resource-constrained scenarios. This demonstrates its suitability for real-time applications, particularly in advanced driver-assistance systems. Future improvements could address misclassification of visually similar signs through enhanced data processing and model refinement.

Conclusion.

This study developed and evaluated a convolutional neural network for traffic sign classification using the GTSRB dataset. The model achieved high accuracy across all stages, from training to testing, proving its efficiency and suitability for practical use. The success of the model can be attributed to balanced data distribution, augmentation techniques, and optimized architecture. Despite minor errors in classifying similar signs, the model showed strong potential for further improvement through additional data and advanced architecture. This work lays the foundation for future research aimed at enhancing the accuracy and reliability of traffic sign classification models, as well as their integration into modern ADAS and autonomous driving systems.

REFERENCE

[1] Tian, Y. (2020). Artificial intelligence image recognition method based on convolutional neural network algorithm. Ieee Access, 8, 125731-125744. https://doi.org/10.1109/ACCESS.2020.3006097

[2] Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2021). Inception recurrent convolutional neural network for object recognition. Machine Vision and Applications, 32, 1-14. https://doi.org/10.1007/s00138-020-01157-3

[3] Hussain, N., Khan, M. A., Sharif, M., Khan, S. A., Albesher, A. A., Saba, T., & Armaghan, A. (2024). A deep neural network and classical features based scheme for objects recognition: an application for machine inspection. Multimedia Tools and Applications, 1-23. https://doi.org/10.1007/s11042-020-08852-3

[4] Zhu, Y., & Yan, W. Q. (2022). Traffic sign recognition based on deep learning. Multimedia Tools and Applications, 81(13), 17779-17791. https://doi.org/10.1007/s11042-022-12163-0

[5] Yang, R., & Yu, Y. (2021). Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis. Frontiers in oncology, 11, 638182. https://doi.org/10.3389/fonc.2021.638182

[6] Jalal, A., Ahmed, A., Rafique, A. A., & Kim, K. (2021). Scene semantic recognition based on modified fuzzy C-mean and maximum entropy using object-to-object relations. IEEE Access, 9, 27758-27772. https://doi.org/10.1109/ACCESS.2021.305898

[7] Uganya, G., Sudha, I., Lakshmanan, V., Shadrach, F. D., Krishnammal, P. M., & Nandhini, T. J. (2023, June). Crime Scene Object Detection from Surveillance Video by using Tiny YOLO Algorithm. In 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN) (pp. 654-659). IEEE. https://doi.org/10.1109/ICPCSN58827.2023.00114v

[8] Ge, H., Zhu, Z., Dai, Y., Wang, B., & Wu, X. (2022). Facial expression recognition based on deep learning. Computer Methods and Programs in Biomedicine, 215, 106621. https://doi.org/10.1016/j.cmpb.2022.106621

[9] Zhang, J., Sun, J., Wang, J., & Yue, X. G. (2021). Visual object tracking based on residual network and cascaded correlation filters. Journal of ambient intelligence and humanized computing, 12, 8427-8440. https://doi.org/10.1007/s12652-020-02572-0

[10] Ainur, J., Gulzhan, M., Amandos, T., Venera, R., Bulat, S., Zauresh, Y., & Aizhan, S. (2024). The impact of blockchain and artificial intelligence technologies in network security for e-voting. International Journal of Electrical & Computer Engineering (2088-8708), 14(6). http://doi.org/10.11591/ijece.v14i6.pp6723-6733.

[11] Мектепбаева, А., Сахипов, А., Рыстыгулова, В., Кайбасова, Д., & Белгибаева, Л. (2024). Optimizing machine learning with quantum enhancements for real-time dynamic systems. Вестник КазАТК, 135(6), 243-254. https://doi.org/10.52167/1609-1817-2024-135-6-243-254.

Айгулим Баегизова, ф.-м.ғ.к., аға оқытушы, Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана, Қазақстан, baegiz_a@mail.ru

Айнур Джумагалиева, магистр, аға оқытушы, Қ.Құлажанов атындағы Қазақ технология және бизнес университеті, Астана, Қазақстан, jumagalievaainur.m.@gmail.com

Венера Рыстыгулова, ф.-м.ғ.к., қауымдастырылған профессор, Қ.Құлажанов атындағы Қазақ технология және бизнес университеті, Астана, Қазақстан, RystygulovaV@gmail.com

Галия Мухамедрахимова, п.ғ.к., аға оқытушы, Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана, Қазақстан, isatai-07@mail.ru

Жанар Ламашева, PhD, аға оқытушы, Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана, Қазақстан, lamasheva_zhb@enu.kz

ЖАСАНДЫ ИНТЕЛЛЕКТТІ НЕЙРОНДЫҚ ЖЕЛІЛЕР НЕГІЗІНДЕ ОБЪЕКТІЛЕРДІ ТАНУ ҮШІН ҚОЛДАНУ

Аңдатпа. Бұл мақала жасанды интеллект пен нейрондық желілерді пайдалану арқылы объектілерді автоматты түрде тану әдістерін әзірлеу мен зерттеуге арналған. Эксперименттік база ретінде әртүрлі объектілердің бейнелерін қамтитын стандартты деректер жиынтығы қолданылды. Бұл модельді жарықтандырудың, көру бұрыштарының және кескін сапасының өзгеруіне төзімді, объектілерді тиімді түрде жіктей алатындай етіп

құруға мүмкіндік берді. Әдістеме деректерді нормализациялау мен аугментациялау, стратификацияланған деректер жиынтығын бөлуді, сондай-ақ есептеу тиімділігі мен жіктеу дәлдігі арасындағы тепе-теңдікті қамтамасыз ету үшін модель архитектурасын оңтайландыруды қамтиды. Әзірленген нейрондық желі қабатты архитектураны, белсендіру функцияларын және тұрақтандыру әдістерін қолданады, бұл модельдің бейімделгіш әрі сенімді болуын қамтамасыз етеді. Зерттеу нәтижелері ұсынылған модельдің жоғары дәлдігі мен сенімділігін көрсетті. Жіктеу қателерін талдау визуалды ұқсас объектілерді тануға қатысты мәселелерді анықтап, оларды азайтуға бағытталған шешімдерді ұсынды. Бұл жұмыс объектілерді тануды автоматтандыруда жасанды интеллект пен нейрондық желілерді қолданудың маңыздылығын атап көрсетіп, олардың интеллектуалды жүйелер мен жоғары дәлдіктегі қосымшаларда қолдану мүмкіндіктерін ашады.

Түйінді сөздер. Жасанды интеллект, нейрондық желілер, тану, жіктеу, бейнелер, нормализация, аугментация, автоматтандыру.

Айгулим Баегизова, к.ф.-м.н., старший преподаватель, Евразийский национальный университет имени Л.Н. Гумилева, Астана, Казахстан, baegiz_a@mail.ru

Айнур Джумагалиева, магистр, старший преподаватель, Казахский университет технологий и бизнеса имени К.Кулажанова, Астана, Казахстан, jumagalievaainur.m.@gmail.com

Венера Рыстыгулова, к.ф.-м.н., ассоциированный профессор, Казахский университет технологий и бизнеса имени К.Кулажанова, Астана, Казахстан, rystygulovav@mail.ru

Галия Мухамедрахимова, к.п.н., старший преподаватель, Евразийский национальный университет имени Л.Н. Гумилева, Астана, Казахстан, isatai-07@mail.ru

Жанар Ламашева, PhD, старший преподаватель, Евразийский национальный университет имени Л.Н. Гумилева, Астана, Казахстан, lamasheva_zhb@enu.kz

ИСПОЛЬЗОВАНИЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ РАСПОЗНАНИЯ ОБЪЕКТОВ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

Аннотация. Данная статья посвящена разработке и исследованию методов автоматического распознавания объектов с использованием искусственного интеллекта и свёрточных нейронных сетей. В качестве экспериментальной базы использовался стандартный набор данных для распознавания дорожных знаков, включающий изображения различных типов объектов, что позволило сосредоточиться на создании модели, способной эффективно классифицировать объекты в различных условиях, включая изменения освещения, углов обзора и качества изображений. Методология включает этапы нормализации и аугментации данных, стратифицированное разбиение выборки, а также оптимизацию архитектуры модели для обеспечения баланса между вычислительной эффективностью и точностью классификации. Разработанная свёрточная нейронная сеть использует слоистую архитектуру с функциями активации и методами регуляризации, что позволяет модели быть устойчивой и адаптивной. Результаты исследования продемонстрировали высокую точность и надёжность предложенной модели. Проведённый анализ ошибок классификации выявил проблемы, связанные с распознаванием визуально схожих объектов, и предложил решения, направленные на их минимизацию. Работа подчёркивает значимость использования искусственного интеллекта и свёрточных нейронных сетей для автоматизации распознавания объектов,

открывая перспективы их применения в интеллектуальных системах и высокоточных приложениях.

Ключевые слова. Искусственный интеллект, нейронные сети, распознавание, классификация, изображения, нормализация, аугментация, автоматизация.

****

Редакцияға түсті / Поступила в редакцию / Received 23.07.2024
Жариялауға қабылданды / Принята к публикации / Accepted 27.12.2024