UDC 004.85

Kumalakov B.*, Abdibek B., Mukhanov D.
Astana IT University, Astana, Kazakhstan
E-mail: b.kumalakov@gmail.com

ABOUT A FILE SORTING PROBLEM USING MACHINE LEARNING

Kumalakov B.A., PhD, ssociate professor at the School of Artificial Intelligence and Data Science, Astana IT University. E-mail: b.kumalakov@gmail.com, ORCID: https://orcid.org/0000-0003-1476-9542;
Abdibek B., Junior researcher. E-mail: ailpokabdibek@gmail.com, ORCID: https://orcid.org/0009-0006-7874-9371;
Mukhanov D., Junior researcher. E-mail: danilmukhanov21@gmail.com, ORCID: https://orcid.org/0009-0003-0753-6553.

File sorting is essential in a wide range of data management areas, including forensic sciences. In some cases, forensics specialists face situations when file extensions are absent or intentionally altered for some purposes. Nevertheless, such files might contain valuable information for the investigation at hand. This paper reports on an attempt to develop a deep learning system to sort files by categories using bit representation of file contents, unlike traditional file extension, header or metadata-based solutions. Training, validation and test datasets included 100 files of various document, media and audio formats. Overall, the system demonstrates accuracy rate that exceeds 95%, with highest inaccuracy when trying to classify files with similar contents (for instance, different video file types, etc.). Nonetheless, the solution serves a solid proof of the concept and a foundation for future research in this direction.

Keywords: machine learning, deep learning, file sorting.

Құмалақов Б.А.*, Әбдібек Б., Мұханов Д.
Astana IT University, Астана, Қазақстан
**E-mail: b.kumalakov@gmail.com*

МАШИНАЛЫҚ ОҚЫТУДЫ ҚОЛДАНА ОТЫРЫП ФАЙЛДАРДЫ СҰРЫПТАУ ЕСЕБІ ТУРАЛЫ

Құмалақов Б.А., PhD, Жасанды интеллект және ерек туралы ғылым мектебінің қауымдастырылған профессоры, Astana IT University. E-mail: b.kumalakov@gmail.com, ORCID: https://orcid.org/0000-0003-1476-9542;
Абдибек Б., кіші ғылыми қызметкер. E-mail: ailpokabdibek@gmail.com, ORCID: https://orcid.org/0009-0006-7874-9371;
Муханов Д., кіші ғылыми қызметкер. E-mail: danilmukhanov21@gmail.com, ORCID: https://orcid.org/0009-0003-0753-6553.

Файлдарды сұрыптау деректерді басқарудың көптеген салаларында, соның ішінде цифрлық криминалистикада кеңінен пайдаланады. Кейбір жағдайларда цифрлық криминалист кейбір мақсаттар үшін файл кеңейтімдері жоқ немесе әдейі өзгертілген жағдайларға тап болады. Дегенмен, мұндай файлдарда тергеу үшін құнды ақпарат болуы мүмкін. Бұл мақалада дәстүрлі файл кеңейтімі, тақырып немесе метадеректер негізінде жұмыс істейтін шешімдерден өзгеше, файл мазмұнының биттік көрінісін пайдаланып файлдарды санаттар бойынша сұрыптау үшін терең оқыту жүйесін әзірлеу әрекеті туралы хабарлайды. Тренинг, валидация және сынақ деректер жинақтары әртүрлі құжат, медиа және аудио форматтағы 100 файлды қамтыды. Жалпы алғанда, жүйе файлдарды 95% дәлдікпен анықтайды, бірақ мазмұны ұқсас файлдарды (мысалы, әртүрлі бейнефайл түрлері және т.б.) анықтағанда қате жіберуі мүмкін. Дегенмен, шешім тұжырымдаманың берік дәлелі және осы бағыттағы болашақ зерттеулер үшін негіз болады.

Түйін сөздер: машиналық оқыту, терең оқыту, файлдарды сұрыптау.

Кумалаков Б.*, Абдибек Б., Муханов Д.
Astana IT University, Астана, Казахстан
**E-mail: b.kumalakov@gmail.com*

ОБ ОДНОЙ ЗАДАЧЕ СОРТИРОВКИ ФАЙЛОВ С ПРИМЕНЕНИЕМ МАШИННОГО ОБУЧЕНИЯ

Кумалаков Б.А., PhD, ассоциированный профессор Школы искусственного интеллекта и Науки о данных Astana IT University. E-mail: b.kumalakov@gmail.com, ORCID: https://orcid.org/0000-0003-1476-9542;
Абдибек Б., младший научный сотрудник. E-mail: ailpokabdibek@gmail.com, ORCID: https://orcid.org/0009-0006-7874-9371;
Муханов Д., младший научный сотрудник. E-mail: danilmukhanov21@gmail.com, ORCID: https://orcid.org/0009-0003-0753-6553.

Сортировка файлов играет важную роль в широком спектре областей управления данными, включая цифровую криминалистику. В некоторых случаях специалисты по цифровой криминалистике сталкиваются с ситуациями, когда расширения файлов отсутствуют или намеренно изменены для каких-либо целей. Тем не менее, такие файлы могут содержать ценную информацию для проводимого расследования. В данной статье представлена попытка разработки системы глубокого обучения для сортировки файлов по категориям с использованием битового представления их содержимого, в отличие от традиционных решений, основанных на расширении файлов, заголовках или метаданных. Обучающие, валидационные и тестовые наборы данных включали 100 файлов различных форматов документов, медиа- и аудиофайлов. В целом система демонстрирует точность, превышающую 95%, с наибольшей погрешностью при классификации файлов со схожим содержимым (например, разных типов видеофайлов и т. д.). Тем не менее, решение служит убедительным подтверждением концепции и основой для будущих разработок в этом направлении.

Ключевые слова: машинное обучение, глубокое обучение, сортировка файлов.

INTRODUCTION

File sorting is essential in a wide range of data management areas [1, 2], including forensic sciences. Of a special interest are cases when experts must deal with damaged files, such as files recovered from damaged memory partitions or – in fact - purposely altered for any reason. Hence, traditional file classification techniques, such as file extension- or header-based approaches, become invalid and use of deep learning systems to examine file fragments - in its binary representation - becomes promising enhancement.

Over past years numerous industrially applicable open-source systems made their way to the market. Namely TrID, DROID, Siegfried, Magika, FiFTy, File Fragment Classification and the "Binary Classification of CSV Files" project gained visibility.

Magika is a software product developed by Google, based on deep neural networks, that classifies over 100 file types with over 99% accuracy [3]. The solution provides fast processing regardless of file size and is available through both a Python command-line interface and an API. Advantages include high classification accuracy, support for a wide range of file types, and ease of integration via the API. However, using Magika requires significant computing resources, and customizing the tool for specific tasks can be complex. FiFTy is a tool that uses neural networks to classify file fragments and supports various scenarios and block sizes, including 512 and 4096 bytes [4]. It supports 75 file types, including archives, documents, media formats, and executable files. FiFTy's advantages include its ability to handle corrupted data and the ability to train on new data. Its disadvantages include the need for pre-training and model training. FiFTy is particularly effective when tailored to specific tasks.

The File Fragment Classification project implements support vector matching (SVM) and k-nearest neighbour (KNN) to classify 13 file types [5, 6, 7, 8]. The solution demonstrates high accuracy on real-world data, making it useful for digital forensics. Advantages also include high accuracy on a limited set of file types

and relatively low computational requirements. However, the project is limited in the number of file types it supports and is less effective when working with fragmented or corrupted data. This solution is suitable for working with small datasets, especially in the presence of partial corruption.

Binary Classification of CSV Files uses various machine learning models for binary classification of CSV files [9], which can be adapted to classify other file types by modifying the dataset and tailoring an algorithm. Key advantages include high accuracy in specific tasks and flexibility in model selection. Limitations include the need to adapt to other file types and a narrow focus on binary classification. This solution can be useful for automating data sorting based on its structural characteristics.

At this stage we can generalise that machine learning enabled tools either rely on availability of an external dependence (such as Goodle database or a library of signatures) or have extremely limited number of file types it can handle. This fact serves as a basis to develop a bespoke solution for forensic file sorting needs this research pursues.

Remainder of this paper reports on a file sorting application which utilises traditional classification technique with an optional deep learning component and its test results. The machine learning based component is a three-layer network trained to recognize 19 (nineteen) file extensions. The system demonstrated 95% classification efficiency with a training time of 6.5 minutes using a graphics processing unit and 30 minutes with central processing unit only. The traditional file classification algorithms include “extensions”, “header” and “signatures” based sorting.

MATERIALS AND METHODS

The workflow is to unpack files stored in a digital archive into target folders according to their type and use. Specifically, category folders are: “media” (with graphics, video, audio and animation sub-folders), “configuration files”, “documents”, “no match” and “others”. Some files are intentionally saved with false extensions (or without one) and some are with stripped metadata.

Schematically algorithm is presented in figure 1. When launched, the program reads the archive and extracts files one by one. If a file is an archive itself, it recursively processes it and carries on till the root archive files remain. In other cases, it tries to classify the file using its “extension”, “header” and/or “signature” in the named order. After that the same file is classified using machine learning and one of three scenarios happen. First, if both methods produce the same outcome, the file is saved into corresponding folder (such as media, document, etc.). Second, if both methods fail to identify file type, it is saved into the “others” folder. Third, if two steps classify the file differently or one of them fails to do so, it is saved to the “no match” folder. Such a naïve approach is designed not to distract the evaluator from machine learning component efficiency and keep experiment setting as simple as possible.

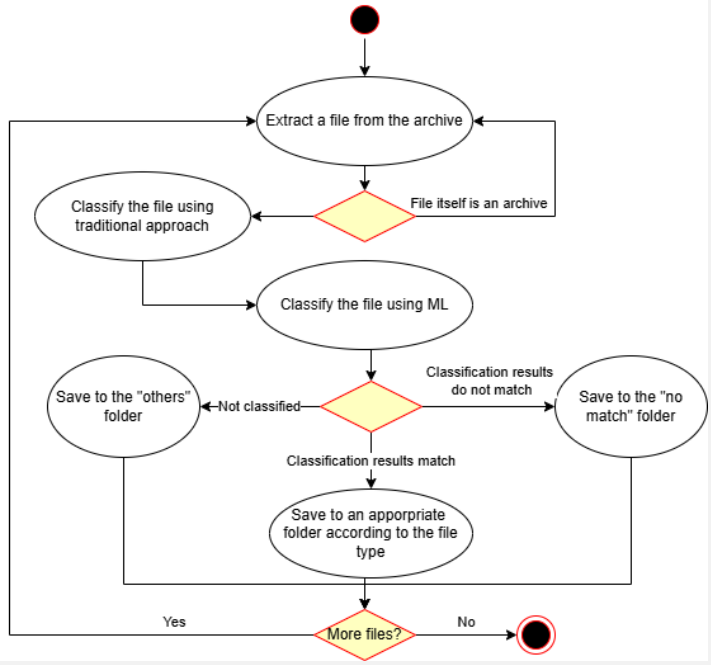

Figure 1 – Schematic representation of the sorting algorithms

Experiment dataset consists of 100 files with the following extensions: doc, docx, html, pdf, pptx, txt, xls, xlsx, xml, gif, jpeg, jpg, png, mov, mp4, audioaac, mp3, and opus. They were proportionally divided into three sets: training (64%), validation (16%), and testing (20%).

For each file, the first 2048 bytes were extracted, of which the first 256 bytes were removed. The remaining 1792 bytes were used as features added to the "content" column. To run the model, binary data was converted into strings. To balance the data, 100 files of each type were randomly selected to avoid imbalance and inaccuracies in the classification of underrepresented file types.

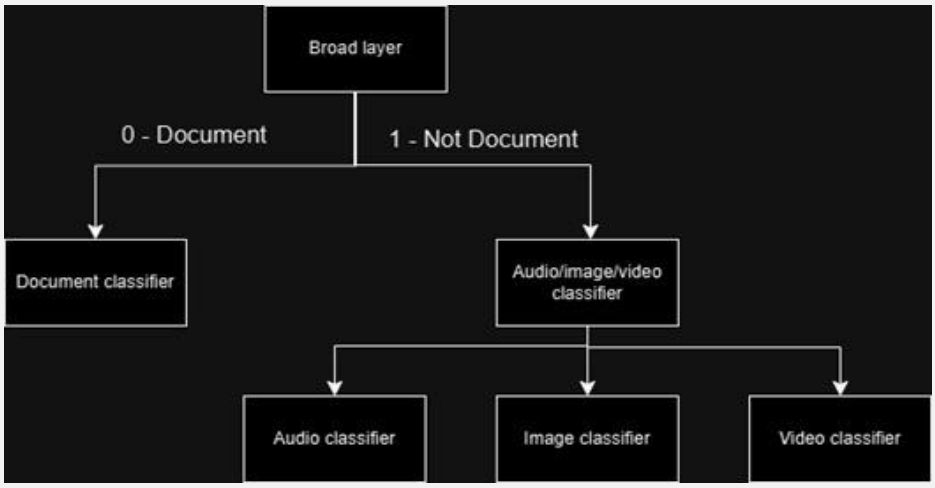

Figure 2 – Deep learning structure

Figure 2 visualises the hierarchical structure of the deep learning system. Every rectangle represents a classifier layer. Various algorithms were investigated, including random forest, support vector machines, XGBoost, and neural networks. The neural network was trained using various hidden layer configurations (convolutional, pooling, and dense) and the number of neurons (from 32 to 256 in each layer). Additionally, the XGBoost classifier was trained on n-gram vectors. Early stopping, dropout, and layer normalization were used to reduce overfitting.

Evaluation metrics included precision and recall, where:

- Precision: the proportion of relevant instances among those retrieved.
- Recall: the proportion of relevant instances that were retrieved.

These metrics were saved in the classification report (see figure 3). The performance was evaluated by file type: precision was analysed for each type to identify anomalies.

```
Audio model classification report:
{'.aac': 0, '.mp3': 1, '.opus': 2, '.wav': 3}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       1.00      0.95      0.97        19
           2       1.00      1.00      1.00        25
           3       0.94      1.00      0.97        15

    accuracy                           0.99        80
   macro avg       0.98      0.99      0.99        80
weighted avg       0.99      0.99      0.99        80
```

Figure 3 – Classification report structure for a single classifier

Forward propagation on the first layer is defined as follows:

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{l}$$

$$a^{(l)} = f(z^{(l)})$$

where $W^{(l)}$– layer weights matrix, $b^{(l)}$ – displacement vector, $a^{(l-1)}$ – output of the previous layer.

Backward propagation is used to calculate gradients and update weights. Gradients are defined using the chain rule:

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l)} \cdot a^{(l-1)}$$

The loss function, in turn, determines the difference between the network predictions and the true values:

$$L = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where m is the number of examples, y_i is the true label, $\hat{y}_i$ is the predicted probability.

$$\delta^{(l)} = \frac{\partial L}{\partial z^{(l)}} \cdot f'(z^{(l)})$$

where $\delta^{(l)}$ — error gradient at the level l.

In turn, the weight update is implemented using the gradient descent method.:

$$W^{(l)} := W^{(l)} - \eta\frac{\partial L}{\partial W^{(l)}}$$

where η is the learning speed.

To implement the algorithm, a convolutional neural network is used, which uses 768 bytes of the input file, totalling 1024 bytes of the input file for processing. The file extension is ignored when reading at the code logic level. The word length of 768 was chosen because a significant proportion of text files (.txt) and animations (.gif) are small, and using longer words would capture part of the file footer.

RESULTS AND DISCUSSION

Deep learning system performance results are presented in figures 4 and 5.

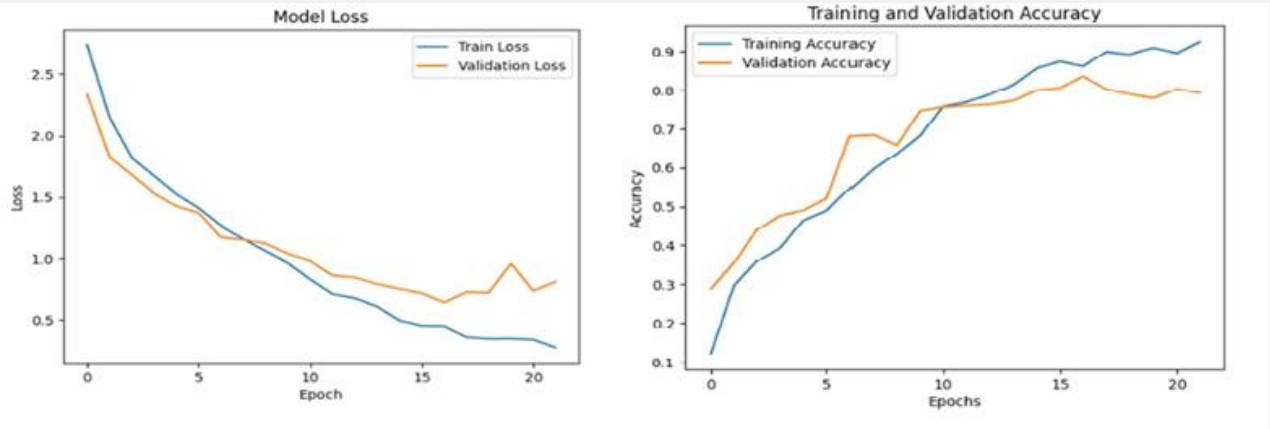

Figure 4 – Model training and validation accuracy

File recognition quality for general categories exceeds 95%. However, test results also indicate that the neural network has difficulty recognizing files with similar content (e.g., png, jpeg, and jpg; as well as audio

files (mp4 and mov). This might be fixed by dynamically increasing the number of input bytes fed to the model for certain file types.

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 249ms/step - accuracy: 0.6041 - loss: 1.6449
Test accuracy for .aac: 0.6299999952316284
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 256ms/step - accuracy: 0.9046 - loss: 0.4458
Test accuracy for .doc: 0.9100000262260437
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 257ms/step - accuracy: 0.9786 - loss: 0.0629
Test accuracy for .docx: 0.9700000286102295
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 253ms/step - accuracy: 0.1579 - loss: 3.9718
Test accuracy for .gif: 0.15000000596046448
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 404ms/step - accuracy: 0.9109 - loss: 0.2106
Test accuracy for .html: 0.9100000262260437
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 249ms/step - accuracy: 0.0554 - loss: 6.0902
Test accuracy for .jpeg: 0.05000000074505806
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 253ms/step - accuracy: 0.5771 - loss: 2.7527
Test accuracy for .jpg: 0.5600000023841858
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 256ms/step - accuracy: 0.1873 - loss: 4.7060
Test accuracy for .mov: 0.20000000298023224
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 263ms/step - accuracy: 0.4122 - loss: 2.2803
Test accuracy for .mp3: 0.38999998569488525
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 247ms/step - accuracy: 1.0000 - loss: 0.0010
Test accuracy for .mp4: 1.0
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 256ms/step - accuracy: 0.9724 - loss: 0.1014
Test accuracy for .opus: 0.9700000286102295
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 253ms/step - accuracy: 0.6190 - loss: 2.1101
Test accuracy for .pdf: 0.6100000143051147
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 262ms/step - accuracy: 0.1359 - loss: 2.6067
Test accuracy for .png: 0.1599999964237213
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 408ms/step - accuracy: 0.8341 - loss: 0.7577
Test accuracy for .pptx: 0.8299999833106995
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 346ms/step - accuracy: 0.9816 - loss: 0.2243
Test accuracy for .txt: 0.9800000190734863
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 259ms/step - accuracy: 0.8892 - loss: 0.4034
Test accuracy for .wav: 0.8999999761581421
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 256ms/step - accuracy: 0.9653 - loss: 0.1291
Test accuracy for .xls: 0.9599999785423279
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 255ms/step - accuracy: 0.9352 - loss: 0.3015
Test accuracy for .xlsx: 0.949999988079071
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 260ms/step - accuracy: 0.9630 - loss: 0.0753
Test accuracy for .xml: 0.9700000286102295
```

Figure 6 – Classification accuracy by file extension

The training was also run using central processing unit only and a graphical processing unit enabled (GPU) modes. When using GPU training took on average 6.5 minutes, while in a CPU only mode it took 30 minutes to reach the same result.

CONCLUSION

Paper presented a hybrid file classification and sorting application that combines traditional algorithms and machine learning techniques to effectively address data management issues for forensics needs. The proposed multi-level classification involves n-gram tokenization, and convolutional neural networks, and achieved classification accuracy exceeding 95% for provided file types, despite the challenges associated with similarity. The use of GPU acceleration significantly reduced training time, making the method scalable and suitable for working with large volumes of data. However, limitations related to the recognition of rare or corrupted file types remain, requiring further research. Future work will focus on improving the model's adaptability, expanding its capabilities to work with more complex datasets, and integrating new technologies to improve accuracy and performance.

Author contributions

Kumalakov B. – principal investigator, manuscript author, solution design, model tuning, assessment design.

Abdibek B. – data preparation, machine learning model implementation and training, programming, building graphs.

Mukhanov D. – programming, performing tests, building graphs.

Information about financing

This study was carried out with the financial support of the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan under Agreement No. 388/PTF-24-26 dated October 1, 2024, under the scientific research project IRN BR24992852 "Development of intelligent models and methods of the Smart City digital ecosystem for sustainable development of the city and improving the quality of life of citizens".

REFERENCES

1. Johnson L.,Williams R. Automation in Data Processing. Journal of Information Systems, 2019. (in English).
2. Smith J. Big Data Management: Strategies and Technologies. International Journal of Data Analysis, 2020. (in English).
3. Google. Magika: Detect File Content Types with Deep Learning. GitHub repository, 2021. URL: https://github.com/google/magika (in English).
4. Mittal G., Korus, P., Memon, N. FiFTy: Large-Scale File Fragment Type Identification Using Convolutional Neural Networks. IEEE Transactions on Information Forensics and Security, 2021, vol. 16, pp. 28-41, DOI: 10.1109/TIFS.2020.3026581. (in English).
5. Cayli M. File Fragment Classification with Machine Learning. GitHub repository, 2017. URL: https://github.com/mervecayli/File_Fragment_Classification (in English).
6. Anil S. Binary Classification of CSV Files using Machine Learning and Deep Learning Models. GitHub repository, 2019. URL: https://github.com/Smitha-anil/Binary_classification (in English).
7. Krasov A.V., Shterenberg S. I., Fakhrutdinov R. M., Ryzhakov D. V., Pestov I. E. Analysis of the information security of an enterprise based on the collection of user data from open resources and monitoring of information resources using machine learning. T-Comm-Telecommunications and Transport, 2018, 12(10), pp. 36-40. DOI: 10.24411/2072-8735-2018-10154. (in Russ.).
8. Matveev A. O., Bystrov A.V., Babaev,V. I., Povarov N. I. Development of software tools to improve the operation of the code auto-completion mechanism using machine learning algorithms in an integrated development environment for the Python language. Bulletin of Novosibirsk State University. Series: Information Technology, 2020, 18(2), pp. 62-75. DOI: 10.25205/1818-7900-2020-18-2-62-75. (in Russ.)
9. Selezneva Ya. M., & Zenkin A.M. Models and methods for ensuring cybersecurity of digital economy systems based on machine learning. In The Almanac of scientific papers of young scientists of ITMO University, 2022, pp. 371-374. URL: https://www.elibrary.ru/item.asp?id=49550578. Accessed: 15.01.2025. (in Russ.)
10. Zhang H., Liang H., Ni T., Huang L., Yang J. (2021). Research on multi-object sorting system based on deep learning. Sensors, 2021, 21(18), 6238. https://doi.org/10.3390/s21186238 (in English).