UDC 004.93'1 http://doi.org/10.15407/pp2023.02.059

Ya. S. Grigorenko, V.M. Shymkovysh, P.I. Kravets, A.O. Novatskyi, L.L. Shymkovysh, A.Yu. Doroshenko

A CONVOLUTIONAL NEURAL NETWORK MODEL AND SOFTWARE FOR THE CLASSIFICATION OF THE PRESENCE OF A MEDICAL MASK ON THE HUMAN FACE

A model of a convolutional neural network, a database for training a neural network, and a software tool for classifying the presence of a medical mask on a person's face, which allows recognizing the presence of a medical mask from the transmitted image, have been developed. The structure of the neural network model was optimized to improve classification results. In addition, the development of the user interface was carried out. The developed application was tested on a set of random images. The resulting model demonstrated high accuracy and robustness in solving the task of classifying the presence of a medical mask on a person's face, which allows automating measures to protect people from the spread of diseases. The implemented application meets the requirements for speed and quality of classification. Further improvement of the classification quality of CNN can be done by collecting a larger dataset and researching other CNN architectures.

Key words: convolutional neural networks, image classification, Python, tensorflow, keras, medical masks.

Introduction

Masks to protect against viruses during the COVID-19 pandemic have become part of the edifice for people around the world. They protect healthy people from infected people. However, since the virus can develop asymptomatically, the obvious solution was the use of masks by all people who are in public places and on the street. Proper use of a medical mask, especially in combination with a vaccine, minimizes the spread of COVID-19, including its new strains. However, the problem arose, how to monitor whether a visitor to a public place complies with the requirement to wear a mask? The state and businesses began hiring workers to control the situation. This is not the best solution for several reasons: increased risks for the employee's health, additional contacts with people in public places, the human factor, it is difficult to provide the required number of employees. This is where modern technologies come to the rescue. By installing a video camera and connecting it to the server, you can track the presence of a medical mask on people's faces. This is possible thanks to neural networks.

Artificial neural networks (ANNs) are mathematical models that imitate the functioning of biological neural networks and are designed to solve various problems in the field of artificial intelligence [1-3]. They are networks of interconnected artificial neurons that process and transmit information among themselves. ANNs can be used in a wide range of applications, such as computer vision, natural language processing, speech recognition, recommender systems, and many others [4-9].

Convolutional neural networks (CNNs) are a special class of artificial neural networks that use convolutional operations in their architecture [10-13]. They were developed specifically for the analysis of visual data and have shown high efficiency in solving computer vision problems. Convolutional networks have become popular due to their properties, such as locality of receptive fields, limited number of parameters, and multi-layeredness, which help them better learn the visual features and structure of images.

ANNs usually contain convolutional layers, activation functions, pooling layers, and fully connected layers. Due to their architecture, convolutional neural networks are able to detect visual patterns at different

levels of abstraction, which allows them to classify and recognize objects in images with high accuracy. ANNs can be used in various applications, such as face recognition, image classification, image segmentation, video analysis, autonomous driving of vehicles, and many other visual tasks.

One of the key factors behind the popularity of convolutional neural networks is their ability to automatically detect and learn features in data without the need for manual feature creation. This contributed to the development of deep learning and accelerated the discovery of new applications for these networks. ANNs have become the foundation of many modern artificial intelligence and machine learning technologies, and their use continues to grow in a wide range of industries and research.

1. Development of CNN

CNN consists of 6 components: input layer, convolutional layers, activation functions, pooling layers, Fully connected layers, output layer. The general structure of a convolutional neural network is shown on fig. 1.

Input layer: the layer at which input data (usually images) is provided for processing by the network.

Convolutional layers: these layers contain a set of filters (kernels) that «collapse» the input data by performing convolutional operations to detect features or image features at different levels of detail.

Activation functions: after each convolutional layer, activation functions (such as ReLU, sigmoid, or tanh) are applied, which add nonlinearity to the model and allow the network to learn more complex functions.

Pooling layers: these layers perform operations of increasing abstraction and reducing the size of data by selecting the most important informational features (for example, using max-pooling or average-pooling).

Fully connected layers: after a sequence of convolutional and pooling layers, data can be passed to fully connected layers that perform classification or regression based on detected features. It is worth noting that before transferring data to fully connected layers, they are usually expanded into a vector.

Output layer: the last fully connected layer gives the output values, which can be represented in the form of a vector of class probabilities, if the problem is classification, or in the form of numerical values, if the problem is regression.

To develop a convolutional neural network, it was decided to use the basic YOLOv4 model, as it is one of the most powerful and effective models for object detection [15, 16]. YOLOv4 is known for its high accuracy and speed, which allows it to be used in real time. These factors determine the choice of this particular model for the development of a convolutional neural network aimed at classifying the presence of a medical mask on a person's face.

The YOLOv4 architecture is based on a combination of several important

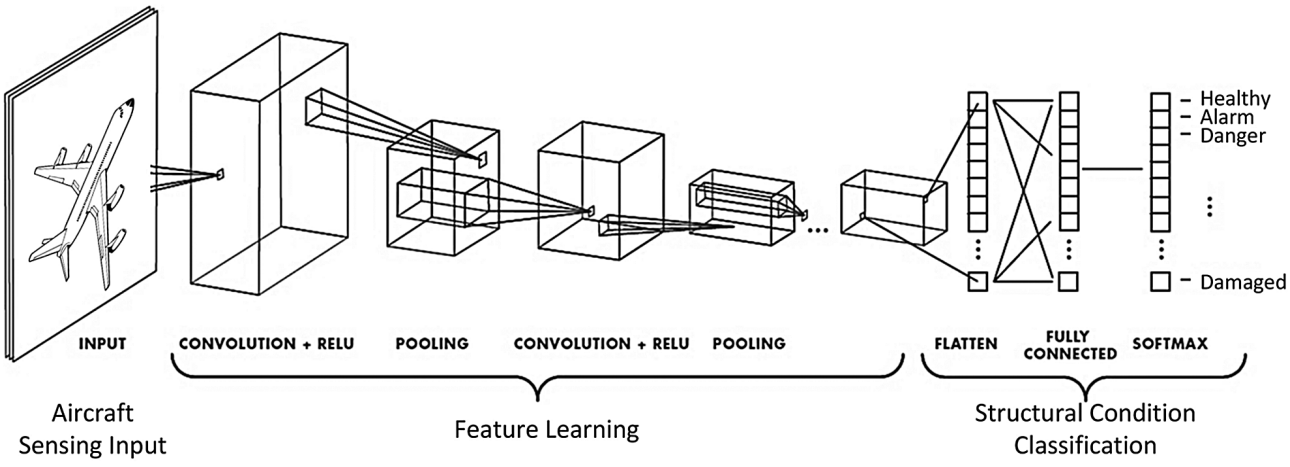

Figure 1. The general structure of a convolutional neural network[14]

components that contribute to its efficiency and accuracy. The main components are:

CSPDarknet53 – basic architecture to ensure high accuracy and relative speed of processing;

Bag of Freebies and Bag of Specials – sets of methods and techniques that take into account various aspects of the network and help improve its speed and accuracy without additional resource costs;

PANet (Path Aggregation Network) is an information aggregation mechanism responsible for combining features at different levels of abstraction to improve object localization accuracy;

SPP (Spatial Pyramid Pooling) is a module that allows the network to take into account contextual information at different scales, which increases the recognition capabilities of the network for objects of different sizes.

To adapt the basic YOLOv4 model to the task of classifying the presence of a medical mask on a person's face, the following studies and improvements have been made. Data collection and annotation: a dataset consisting of photographs of people with and without medical masks was collected. The data were annotated to train the models. Fine-tuning of the model: the technique of fine-tuning was applied to adapt the YOLOv4 model to the specifics of the task. The learning process included setting hyperparameters such as learning rate, batch size, number of epochs, and regularization, which were adjusted to achieve optimal accuracy and learning speed. Also, the learning process included optimization of the architecture. In particular, the number of output classes was changed taking into account the needs of the task.

Also, the sizes of convolutional filters and the sizes of kernels were revised to match the sizes of the faces depicted in the photographs. Evaluation of the results: the model was tested on the test data set and the results were evaluated using metrics such as accuracy, precision, recall and F1-score. Making improvements: after fine-tuning and evaluating the model results, several possible directions for optimization and refinement of the model were identified. These improvements are aimed at increasing the accuracy and efficiency of the model in the task of classifying the presence of a medical mask on a person's face. One of the directions is the modification of convolutional layers: the influence of the number and size of nuclei in convolutional layers on the accuracy and speed of model learning was analyzed. Based on the analysis, it was decided to change some parameters of the convolutional layers, which helped to improve the overall performance of the model and its adaptation to the task of classifying the presence of a medical mask.

2. Development of a dataset for the training of CNN

Data collection and preparation is an important stage in the process of developing and training a convolutional neural network [17-19]. This section describes the data collection process for training the YOLOv4 model adapted to classify the presence of a medical mask on a human face.

To ensure the representativeness of the data and the variety of scenarios, images of people with different types of masks, as well as without masks, were collected. Images have been collected from various sources such as public databases, open websites, and by taking photos in real-world settings. The total volume of the collected dataset is 5 thousand images depicting people of different ages, genders and ethnicities.

After the images were collected, each one was annotated, which included identifying and labeling the face region and the corresponding class (masked or unmasked person). Annotation was performed manually using a specialized software tool that allows to create rectangular frames around the face area and assign them appropriate class labels. The marking process is shown in fig. 2.

In addition to data collection and preparation, data augmentation was applied to provide a greater variety of images and strengthen the robustness of the model [20-22]. Some of the augmentation techniques that were used in this study include scaling, rotation, panning, and illumination.

Scaling – images are scaled to different sizes, which allows the model to learn to recognize objects of different sizes.

Figure 2. Labeling of the dataset

Rotation – images are rotated to arbitrary angles, which allows the model to better adapt to different orientations of objects.

Shift – images are shifted up, down, left or right, which helps the model learn to recognize objects in different positions in the image.

Horizontal display – images are displayed horizontally, which helps the model adapt to symmetry and asymmetry of objects.

Change in illumination – the intensity of illumination in the image changes, which allows the model to better adapt to different lighting conditions. The collected and annotated images were divided into training, validation and test datasets according to the standard 70/15/15 percentile distribution principle.

To ensure the reliability and accuracy of the model, it was important to ensure a balance of classes in the training data set. This means that the number of images with and without masks was about the same.

After data collection, annotation, augmentation, and segmentation, additional data preparation was performed to train the model. In particular, the images were scaled to the same size corresponding to the input size of the YOLOv4 model, and the pixel values were normalized. This helps provide optimal conditions for model training and ensures better utilization of computing resources during the training process.

As a result of data collection and preparation, the obtained dataset was used for training, validation and testing of the YOLOv4 model, adapted for the classification of the presence of a medical mask on a person's face.

3. Description of application implementation

The developed application for recognizing a medical mask on a person's face consisted of 2 modules: a module for providing a graphical interface; image classification and processing module.

The purpose of the first module is to provide a full-fledged graphical interface consisting of the main page, on which there is a component for displaying the video stream from the video camera in real time with the result of the work of the classification module.

The image classification and processing module is the core of the developed application. This module consists of three main components: loading and preparing the image, directly classifying the image, and overlaying the results on the

original image. The classification component uses a trained ANN model based on YoloV4 to classify pre-processed images coming from a video stream. The image preparation component provides functionality that normalizes images and performs the necessary transformations. This module was implemented in the Python programming language. Tensorflow and Keras libraries were used to work with neural networks [23-27]. For image processing – OpenCV and Pillow libraries [28,29]. The Numpy library was used to work with multidimensional arrays.

The code of the CNN model in Python is given in Fig. 3.

```
#implementation Yolov4
def YOLOv4(input_layer):
    route_1, route_2, conv = cspdarknet53(input_layer)

    route = conv
    conv = convolutional(conv, (1, 1, 512, 256))
    conv = upsample(conv)
    route_2 = convolutional(route_2, (1, 1, 512, 256))
    conv = tf.concat([route_2, conv], axis=-1)

    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))

    route_2 = conv
    conv = convolutional(conv, (1, 1, 256, 128))
    conv = upsample(conv)
    route_1 = convolutional(route_1, (1, 1, 256, 128))
    conv = tf.concat([route_1, conv], axis=-1)

    conv = convolutional(conv, (1, 1, 256, 128))
    conv = convolutional(conv, (3, 3, 128, 256))
    conv = convolutional(conv, (1, 1, 256, 128))
    conv = convolutional(conv, (3, 3, 128, 256))
    conv = convolutional(conv, (1, 1, 256, 128))

    route_1 = conv
    conv = convolutional(conv, (3, 3, 128, 256))
    conv_sbbox = convolutional(conv, (1, 1, 256, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

    conv = convolutional(route_1, (3, 3, 128, 256), downsample=True)
    conv = tf.concat([conv, route_2], axis=-1)

    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))

    route_2 = conv
    conv = convolutional(conv, (3, 3, 256, 512))
    conv_mbbox = convolutional(conv, (1, 1, 512, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

    conv = convolutional(route_2, (3, 3, 256, 512), downsample=True)
    conv = tf.concat([conv, route], axis=-1)

    conv = convolutional(conv, (1, 1, 1024, 512))
    conv = convolutional(conv, (3, 3, 512, 1024))
    conv = convolutional(conv, (1, 1, 1024, 512))
    conv = convolutional(conv, (3, 3, 512, 1024))
    conv = convolutional(conv, (1, 1, 1024, 512))

    conv = convolutional(conv, (3, 3, 512, 1024))
    conv_lbbox = convolutional(conv, (1, 1, 1024, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

    return [conv_sbbox, conv_mbbox, conv_lbbox]
```

Figure 3. Code of the CNN model in Python.

4. Results of work and testing

The learning outcomes of the developed ZNM model are as follows:

1. Accuracy of image classification on the training set: 97.3%;

2. Accuracy of image classification on the test set: 92.5%;

3. The average training time in the era is 27 minutes.

Results of testing the average number of frames per second: the speed of image classification by trained SNM is 20-25 frames per second.

The results of testing the application satisfy the technical requirements for its operation. An example of the test result can be seen in fig. 4 and fig. 5.

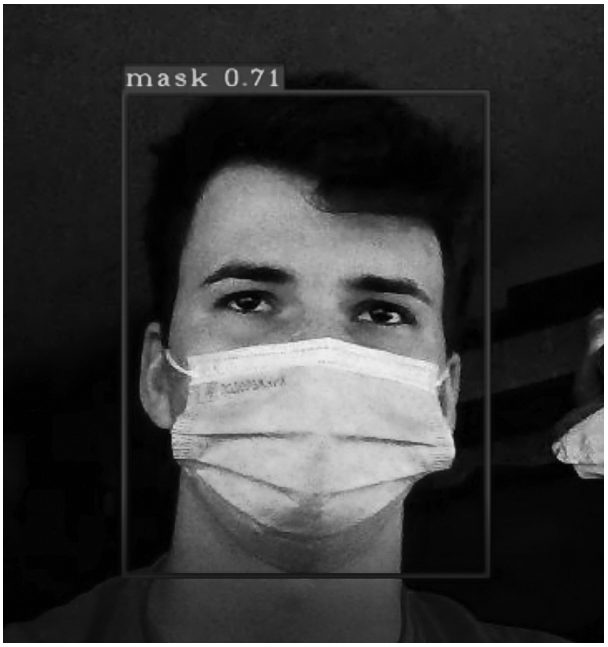

Figure 5. The result of testing with a mask

Conclusions

In this study, a convolutional neural network based on the YOLOv4 architecture was developed to classify the presence of a medical mask on a human face. The fine-tuning technique was applied to adapt the model to the specifics of the task, and the architecture and learning hyperparameters were optimized.

A large dataset with different types of masks and without masks was collected and annotated to successfully train and validate the model. The use of data augmentation provided a greater variety of images and

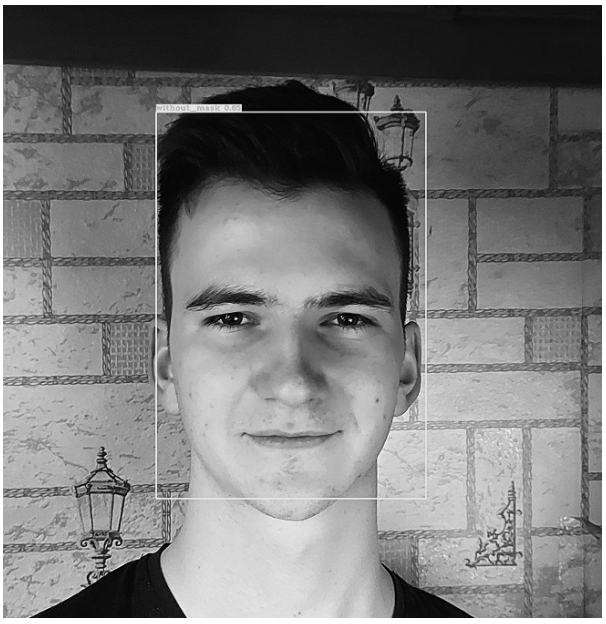

Figure 4. The result of testing without a mask

improved the robustness of the model.

The resulting model demonstrated high accuracy and robustness in solving the task of classifying the presence of a medical mask on a person's face, which allows automating measures to protect people from the spread of diseases.

The implemented application meets the set requirements for speed and quality of classification.

Further improvement of the classification quality of CNN can be done by collecting a larger dataset and researching other CNN architectures.

References

1. Dreyfus G (2005) Neural networks: methodology and applications. Springer-Verlag, Berlin. https://doi.org/10.1007/3-540-28847-3
2. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE.(2017) A survey of deep neural network architectures and their applications. Neurocomputing. vol. 234, pp. 11-26.
3. Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, Shyu ML, Chen SC, Iyengar S. (2018) A survey on deep learning: algorithms, techniques, and applications. ACM Comput Surv (CSUR).vol. 51, no. 5, pp. 1-36.
4. Z. -Q. Zhao, P. Zheng, S. -T. Xu and X. Wu. (2019) Object Detection With Deep Learn-

ing: A Review. IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232. https://doi.org/10.1109/TNNLS.2018.2876865
5. Shymkovych V., Telenyk S., Kravets P. (2021) Hardware implementation of radial-basis neural networks with Gaussian activation functions on FPGA. Neural Computing and Applications. vol. 33, no.15, pp. 9467-9479. https://doi.org/10.1007/s00521-021-05706-3
6. Tian H, Chen SC, Shyu ML.(2020) Evolutionary programming based deep learning feature selection and network construction for visual data classification. Inf Syst Front, vol. 22, no. 5, pp. 1053-1066.
7. Bezliudnyi Y., Shymkovysh V., Doroshenko A.(2021) Convolutional neural network model and software for classification of typical pests. Prombles in programming. vol.4, pp. 95-102. https://doi.org/10.15407/pp2021.04.095
8. Kravets P., Nevolko P., Shymkovych V., Shymkovych L. (2020) Synthesis of High-Speed Neuro-Fuzzy-Controllers Based on FPGA. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT). pp. 291-295. https://doi.org/10.1109/ATIT50783.2020.9349299
9. Shymkovych, Volodymyr, Anatoliy Doroshenko, Tural Mamedov, and Olena Yatsenko (2022) Automated Design of an Artificial Neuron for Field-Programmable Gate Arrays Based on an Algebra-Algorithmic Approach. International Scientific Technical Journal «Problems of Control and Informatics» vol. 67, no. 5, pp. 61-72. https://doi.org/10.34229/2786-6505-2022-5-6
10. Dhillon A, Verma GK. (2020) Convolutional neural network: a review of models, methodologies and applications to object detection. Prog Artif Intell. vol.9, no. 2, pp. 85-112.
11. Khan A, Sohail A, Zahoora U, Qureshi AS. (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev. vol. 53, no. 8, pp. 5455-5501
12. Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, vol. 8, no. 53, pp. 1-74. https://doi.org/10.1186/s40537-021-00444-8
13. Khan, A., Sohail, A., Zahoora, U. et al. (2020) A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, vol. 53, pp. 1-62. https://doi.org/10.1007/s10462-020-09825-6
14. Tabian I, Fu H, Sharif Khodaei Z. A. (2019) Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. Sensors, vol. 19, №22:4933. https://doi.org/10.3390/s19224933
15. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.
16. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.
17. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in neural information processing systems (pp. 1097-1105).
18. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
19. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
20. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
21. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), 1-48.
22. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In International Conference on Machine Learning (pp. 6105-6114).
23. TensorFlow. (n.d.). TensorFlow: An end-to-end open source machine learning platform. Retrieved from https://www.tensorflow.org/
24. Keras. (n.d.). Keras: The Python deep learning API. Retrieved from https://keras.io/
25. Redmon, J. (n.d.). Darknet: Open source neural networks in C. Retrieved from https://pjreddie.com/darknet/
26. COCO Dataset. (n.d.). Common Objects in Context. Retrieved from https://cocodataset.org/

27. ImageNet. (n.d.). ImageNet: A large-scale hierarchical image database. Retrieved from http://www.image-net.org/
28. OpenCV. (n.d.). OpenCV: Open source computer vision library. Retrieved from https://opencv.org/
29. Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (n.d.). Albumentations: Fast image augmentation library. Retrieved from https://github.com/albumentations-team/albumentations

Received: 28.04.2023

Про авторів:

Григоренко Ярослав Сергійович,
студент 4 курсу Національного Технічного Університету України «КПІ імені Ігоря Сікорського».

Шимкович Володимир Миколайович,
кандидат технічних наук,
доцент кафедри інформаційних систем та технологій Національного Технічного Університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 30.
Кількість наукових публікацій в зарубіжних виданнях – понад 10.
Індекс Хірша – 4. https://orcid.org/0000-0003-4014-2786

Новацький Анатолій Олександрович,
кандидат технічних наук,
доцент кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 30.

Кравець Петро Іванович,
кандидат технічних наук,
доцент кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 40. Кількість наукових публікацій в зарубіжних виданнях – понад 10.
Індекс Хірша – 4. https://orcid.org/0000-0003-4632-9832

Шимкович Любов Леонідівна,
асистент кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – 2.
Кількість наукових публікацій в зарубіжних виданнях – 1. https://orcid.org/0000-0002-1291-0373

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії комп'ютерних обчислень,
професор кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 200. Кількість наукових публікацій в зарубіжних виданнях – понад 90.
Індекс Хірша – 6. http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», проспект Перемоги 37 та
Інститут програмних систем НАН України, 03187, м. Київ-187, проспект Академіка Глушкова, 40.
E-mail:
yarik13371337@gmail.com,
v.shymkovych@kpi.ua,
a.novatskyi@kpi.ua,
peter_kravets@yahoo.com,
L.shymkovych@gmail.com,
doroshenkoanatoliy2@gmail.com