UDC 004.93'1 http://doi.org/10.15407/pp2021.04.95

Y. Bezliudnyi, V. Shymkovych, A. Doroshenko

A MODEL OF CONVOLUTIONAL NEURAL NETWORK AND A SOFTWARE APPLICATION FOR TYPICAL INSECT PESTS RECOGNITION

A model of a convolutional neural network, a dataset for neural network training, and a software tool for the classification of typical insect pests have been developed, which allows recognizing the class of insect pests from an image. The structure of the neural network model was optimized to improve the classification results. In addition, the user interface, authentication, and authorization, data personalization, the presence of user roles and the appropriate distribution of functionality by role, the ability to view statistics on classified insects in a certain period of time were developed. Functional testing of the developed software application on a heterogeneous set of images of insects of 20 different classes was performed.
Keywords: image classification, convolutional neural networks, modular architecture, Python, Keras

Introduction

Applied entomology is a branch of biology that studies insects that damage plants, forests, crop products, and studies methods of insect pests control. Applied entomology is given considerable attention in the context of agronomy, which is quite rational, because the wrong definition of the type of insect pests on the field area, the choice of the wrong method of insect pests control or application of crop protection untimely cause a significant decline in yields and therefore loss of farmers. Classification of insects on plantations, in order to use appropriate methods of protection, as well as the collection of statistical data on the quantitative characteristics of crop damage by insect pests, is quite a difficult task, especially in a situation when different types of crops are grown on the single plantation. In addition, many species of insects migrate from one territory to another, some of the species are quite similar to each other, insects can damage plantations at several different stages of development. All of the above complicates the task of correct classification of insect pests by humans.

Artificial neural networks are an attempt to transfer a set of biological mechanisms occurring in the brain of animals to the category of computer systems [1]. Artificial neural networks can be used in facilities such as robotics [2], vehicle and unmanned aerial vehicle control, pattern recognition, analysis and decision making in IoT systems, spacecraft control, military equipment, and many other applications in modern technologies [3-5]. In these systems neural networks can be used for objects identification, prediction of the state of objects, recognition, clustering, classification, analysis of large amounts of data coming at high speed from a large number of devices and sensors, etc. [6-14]. Tasks that involve image classification are mostly based on the use of convolutional neural network technology. A convolutional neural network (CNN) is a type of multilayer perceptrons designed to minimize the amount of pre-processing of input data [15]. This type of neural network architecture has become popular as a tool for solving image recognition problems, as it has several advantages over other architectures, such as fewer learnable parameters; absence of image pixels memorization; the ability to perform part of the calculations in parallel; relative resistance to rotation and shift of images [16].

The purpose of this work is to automate the process of recognizing insect pests through the development of a new model of CNN, as well as learning database creation and the development of software application based on this model.

1. CNN model development

The CNN, in the general case, consists of 5 basic types of layers: input layers, convolutional layers, pooling layers, fully-con-

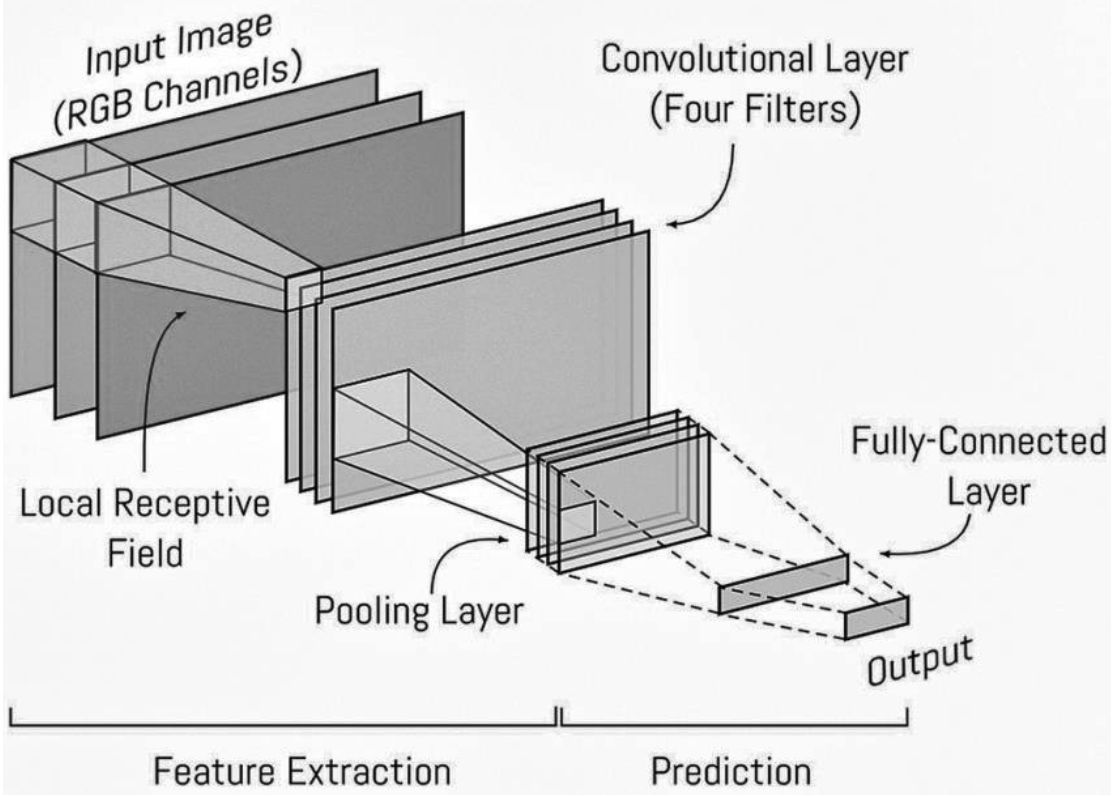

Fig. 1. The general structure of the CNN

nected layers and output layers. The general structure of the CNN is shown in Figure 1.

The input data is an image of $N \times N$ pixels and with K channels of the color model. The input layer of the CNN is a matrix of size $N \times N \times K$, each element of which is a numerical representation of the pixel regarding a particular color channel.

Convolutional layers consist of a set of feature maps, each of which has a scanning kernel (filter). The kernel is a system of scales (synapses). The kernel slides across the map of features and detects certain features of objects. During sliding on the feature map, the kernel performs a convolution operation on the map elements, the result of which is entered into the corresponding cell of the original feature map of a particular convolution layer. The convolution formula is as follows:

$$(A \circ G)[m,n] = \sum_{k,l} A[m-k,n-l] * G[k,l]$$

where A – the matrix of elements of the input feature map; G – the core of the convolution.

In this case, depending on the method of processing the edges of the output matrix, the output map of features may be less than, equal to, or greater than the input. According to the above, the convolutional layer, in a simplified form, is described by the following formula:

$$Y_{k,j} = s\left(\sum_i Y_{k-1,i} \circ G_{k,j} + B_{k,j}\right)$$

where $Y_{k,j}$ – the output of the k-th convolutional layer by the j-th feature; $B_{k,j}$ – the bias of the k-th convolutional layer by the j-th feature; $G_{k,j}$ – the kernel of the k-th convolutional layer by the j-th feature; s – activation function.

Unlike a standard neural network, each synapse in the convolutional layer of the CNN is not associated with all the synapses of the previous layer but is simply connected to nodes in a special region known as the local receptive field.

The pooling layer is usually used after the convolution layer. This type of layer was designed to simplify information and reduce the scale of feature maps made by convolutional layers. In other words, pooling layers create a compressed feature map from each feature map in the convolutional layers. Pooling layers are needed to reduce the time of computations and resolve problems with overtraining of the CNN. The pooling

operation can be performed in different ways, such as a geometric mean, harmonic mean, or maximum of passed arguments. Maximum pooling and average pooling are the two most common pooling operations.

Formally, the pooling layer can be described by the following formula:

$$Y_k = s(P(Y_{k-1}) + B_k)$$

where Y_k – the output of the k-th pooling layer; B_k – the bias of the k-th pooling layer; $P(...)$ – pooling operation; s – activation function.

Fully-connected layers are arranged after a sequence of convolutional layers and pooling layers. This part of the CNN contains a set of critical information obtained from all previous procedures performed inside of the CNN. The neurons of each map of the pooling layer (or convolutional layer, if it is the last before the fully-connected layers) are connected to one neuron of the fully connected layer.

The outputs of the last of the fully-connected layers form the output layer, which is known as the classifier. It determines the probability distribution of each label over N classes. The size of the last layer is equal to the number of classes that are classified by the neural network.

As part of this study, a number of decisions were made regarding the choice of CNN architecture, the algorithms used for input data processing as well as CNN training and interpretation of source data.

The first layer of the CNN is the input layer.

Input data must be normalized before processing by the CNN. For each specific pixel of the input image is its normalization in the range from 0 to 1 by the formula:

$$f(p, min, max) = \frac{p - min}{max - min}$$

where p – value of pixel; min – minimal value of pixel; max – maximal value of pixel. Images are passing to CNN in JPEG format with a RGB color model, which contains 3 channels with pixel values for a specific channel in the range from 0 to 255. Chosen image size is 100×100. This image size is sufficient to classify insect pests.

The input layer is followed by five convolutional layers, between which there are four pairs of normalization and pooling layers.

The size of the kernel in the convolutional layers is usually taken in the range from 3×3 to 7×7. If the size of the kernel is too small, CNN will not be able to distinguish any features; if it is too large then the number of connections between neurons increases. Also, the size of the kernel should be chosen in a way to make the size of the output maps even, which allows to not lose information when reducing the size of the data in the pooling layer described below.

The maximum pooling was chosen as the pooling operation, as it allows to better highlight sharp areas of the image, which is necessary to solve the problem posed.

In practice, the number of fully-connected layers is chosen not too big, usually no more than three. In this paper, it was chosen equal to three, which was determined by empirical method. Two dropout layers were also added between the fully-connected layers to avoid network overtraining.

The Adam algorithm [2] was chosen as an algorithm for gradient descent optimization, which was developed specifically for learning of deep neural networks. The Adam algorithm uses the power of adaptive learning methods that find individual learning metrics for each neural network parameter. Formally, this algorithm is described as follows:

1. Calculate the first and second-order moments of the gradient:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where m_t – the first-order moment of the gradient with regard to model parameter θ_t; v_t – the second-order moment of the gradient with regard to model parameter θ_t; g_t – gradient value with regard to model parameter θ_t.

2. Calculate bias-corrected moments of the gradient:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

3. Update the values of model parameters:

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t$$

where μ – learning rate; ε – small constant.

The authors of the Adam algorithm suggest the use of the following coefficient values: $\beta_1 = 0.9;\ \beta_2 = 0.999;\ \varepsilon = 10^{-8}$.

The sparse categorical cross-entropy was chosen as a function of learning losses [17]. Cross-entropy losses determine the efficiency of the classification model, the output of which is the probability value in the range from 0 to 1. These losses increase when the predicted probability deviates from the actual label. The formula for cross-entropic losses is defined as:

$$Loss = -\sum_{c=1}^{M} y_{o,c} \ln(p_{o,c}),$$

where M – number of classes; y – binary indicator (0 or 1) that shows whether the label c is the correct classification for observation o; $p_{o,c}$ – the assumed probability that the observation o belongs to the class c.

The only difference between sparse categorical cross-entropy and categorical cross-entropy is the label format.

Each layer of the neural network has inputs with a corresponding probability distribution, which in the learning process is influenced by the randomness of the initialization of parameters and randomness in the input data. The influence of these sources of randomness on the probabilistic distribution of the values of the input data on the inner layers during training is described as the internal covariate shift. During the network learning phase, as the parameters of the previous levels change, the probability distribution of the input data to the current level changes accordingly, so that the current level requires constant readjustment to the new distributions. This problem is especially serious for deep networks, as small changes in the hidden layers closer to the input layer will be amplified when they propagate within the network, which will lead to a significant shift in the more distant hidden layers. Therefore, a method of batch normalization [18] has been proposed to reduce these undesirable changes, speed up learning and make the CNN more reliable. Formally, the batch normalization algorithm is defined as follows:

1. The mean μ_B and the variance σ_B^2 of a batch of m elements are computed:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

2. Normalization of layer inputs is performed:

$$\overline{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

where x_i – value of the i-th element of normalization input layer; $\overline{x_i}$ – normalized value of the i-th element of normalization input layer;

ε – small constant.

To restore the representation power of the network transformation of normalized values performed:

$$y_i = \gamma x_i + \beta$$

where y_i – value of the i-th element of normalization layer output; γ, β – parameters that are subsequently learned in the optimization process.

In addition to reducing the internal covariate shift, group normalization provides many other benefits. With using batch normalization technique, the network can use a higher learning rate without problem of gradients values vanishing or exploding.

2. Creating a learning dataset for CNN

The first step in creating a dataset for CNN learning was to determine the classes of insects that will be recognized and the minimum number of image examples in each class. As result, 20 most typical classes of insect pests were selected. The requirement to have at least 40 different images of insects of each class was set. In order to reduce the possibility of the CNN overtraining on the spatial location of insects in the images, for each original image in the database was added 3 more copies with different angles of rotation relative to the original (90°, 180°, 270°). After that images of each class were divided into training and test samples, in a ratio of 2:1.

Note that if the image size is too large, the computational complexity will increase, respectively, the restrictions on the speed of response will be violated. If too small images will be chosen, then the network will not be

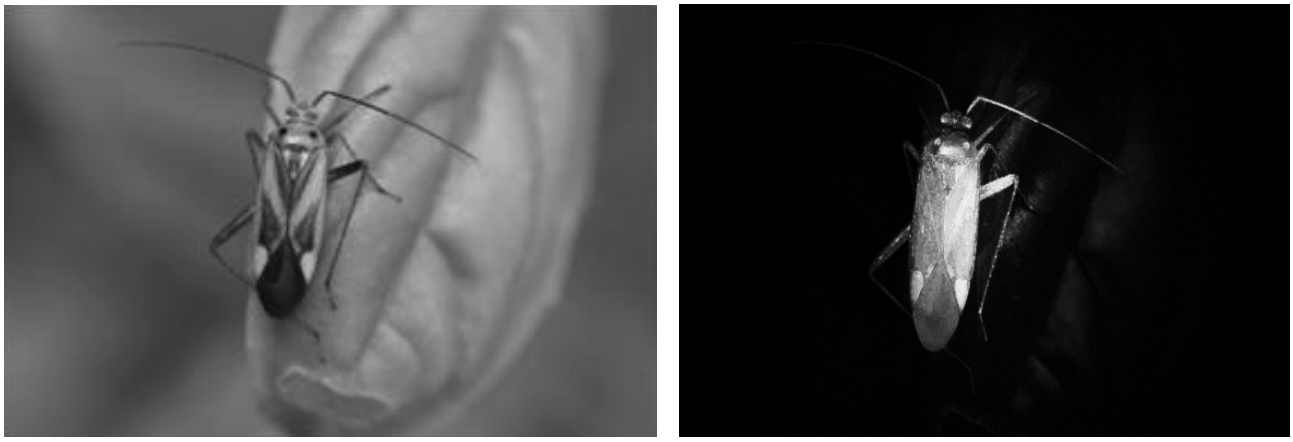

Fig. 2. Image of an insect (on the left) and its saliency map (on the right)

able to detect the key features of the objects depicted on them. In our case, the optimal image size was 100 × 100 pixels.

The dataset was partially populated with images from the IP102 dataset, which contains images of 102 classes of insects, with a total size of 75,000 images. However, most of the insect classes in this set not belong to the category of pests, so the rest of the classes were filled with images obtained through the use of Flickr API. Since the developed CNN is designed to recognize insects in images, the background information is irrelevant. Therefore, in order to improve the accuracy of classification, before scaling and expanding the number of images, an operation was performed to remove excess parts of the image using saliency maps [19]. The purpose of the saliency map is to display the degree of "informativeness" of each pixel, which is respectively greater for the pixels belonging to the part of the image where the insect is located, and less for the background pixels. An example of a saliency map is shown in Figure 2. During the learning dataset preparation, based on the saliency map obtained from the image, the excess parts of the images were cropped. The cropping process started from the image edges until the total "informativeness" of the pixels at the edge of an image passes the specified threshold. Testing on the selected set of images shows that the use of saliency maps allows to increase the insect to background ratio by an average of 30%.

In total, the created dataset contains 3000 images, which are arranged hierarchically by directories, according to their class and purpose (for training or testing).

3. Software application development

The developed application for insect pests recognition consists of three modules:

1. The graphical user interface module;
2. The module of customer accounting and service;
3. The module of image classification and processing.

The purpose of the first module is to provide a full graphical user interface consisting of a home page, pages for registration and login, a page for manual classification, a page for downloading new images of insects (for system administrators), and a page for viewing personal statistics. This module was implemented on the basis of Angular framework, using TypeScript language.

The module of customer accounting and service deals with such tasks as customer-related information processing and customer authorization and authentication. To implement this module, it was decided to choose Java programming language and frameworks such as Spring Security, Spring Boot, Spring Data.

The module of image classification and processing is the core of the developed application. This module consists of two main layers: the layer of classification and the layer of image processing and loading. The classification layer uses a trained CNN model to classify insects in the images that come with the request (images are pre-processed using saliency maps and scaled). The layer of image processing provides functionality that allows making such image transformations as rotation, scaling, highlighting the main part (using saliency maps), and so on. This module was implemented in Python

programming language. Tensorflow and Keras libraries were used to work with neural networks. OpenCV and Pillow libraries were chosen for imageprocessing. Numpy library was used to work with multidimensional arrays.

CNN implementation code in Python:

```
def get_model(image_size, classes_
num):
model = keras.Sequential([
keras.Input(image_size),  # Input layer
        keras.layers.experimental.preprocess-
ing.Rescaling(scale=1/255.0),  # Image scaling
        keras.layers.Conv2D(filters=16,
kernel_size=(7, 7), activation='relu',
padding='same'),  # Convolutional layer
        keras.layers.BatchNormalization(),  #
Normalization layer
        keras.layers.MaxPooling2D(pool_
size=(2, 2), strides=2),  # Pooling layer
        keras.layers.Conv2D(filters=32,
kernel_size=(5, 5), activation='relu',
padding='same'),
        keras.layers.BatchNormalization(),
        keras.layers.MaxPooling2D(pool_
size=(2, 2), strides=2),
        keras.layers.Conv2D(filters=32,
kernel_size=(3, 3), activation='relu',
padding='same'),
        keras.layers.BatchNormalization(),
        keras.layers.MaxPooling2D(pool_
size=(2, 2), strides=2),
        keras.layers.Conv2D(filters=64,
kernel_size=(3, 3), activation='relu',
padding='same'),
        keras.layers.Flatten(),  # Layer dimen-
sionality reduction
        keras.layers.Dense(units=2048,
activation='relu'),  # Fully-connected layer
        keras.layers.Dropout(rate=0.5),  #
Dropout layer
        keras.layers.Dense(units=2048,
activation='relu'),
        keras.layers.Dropout(rate=0.5),
        keras.layers.Dense(classes_num,
activation='softmax')  # Output layer
    ])
        return model
```

CNN training code in Python:

```
        def train_model(
        dataset_path=CURR_DATASET_
PATH,  # Path to the dataset root folder
        write_path=CURR_MODEL_PATH,  #
Path to the pretrained CNN model
    lr=0.0001,  # Learning rate
    epochs_num=15,  # Number of learning ep-
ochs
    image_size=(100, 100, 3),  # Image size
classes_num=20  # Number of classified classes
):
        training_data = keras.preprocessing.
image_dataset_from_directory(dataset_
path + '/train', batch_size=32, image_
size=image_size[:2])
    test_data = keras.preprocessing.image_da-
taset_from_directory(dataset_path + '/test',
batch_size=32, image_size=image_size[:2])
    if (os.path.exists(write_path)): shutil.
rmtree(write_path)
        model = get_model( image_
size=image_size, classes_num=classes_num)
        model.compile( optimizer=keras.opti-
mizers.Adam(lr=lr),
        loss=keras.losses.SparseCategorical-
Crossentropy(),
        metrics=['accuracy'])
    model.fit(training_data, epochs=epochs_
num, verbose=2)
    model.evaluate(test_data, verbose=2)
    model.save(write_path)
```

Optimal values of parameters such as kernel size and number of filters for convolutional layers, number of neurons for fully-connected layers, dropout value and learning rate are based on general recommendations for CNN construction and training described in section 2, as well as on values got using the Keras Tuner library, which provides ready-made solutions for neural network parameter optimization problems.

In addition to the modules described above, the structure of the software solution also includes the PostgreSQL database, which is accessed by the module of accounting and customer service.

4. Software application testing

PC characteristics on which the application was tested:

1. Processor clock speed – 2.6 GHz;
2. 6 physical cores, 12 logical processors;
3. 16 GB of RAM.

4. GPU integration absent.

The following technical requirements were set for the development system:

Accuracy of image classification –

not less than 95% on the training dataset, and not less than 60% on the test dataset;

2. The number of epochs of learning the neural network – 15;

3. The average image classification time – less than 0.5 s;

4. The average download time of the image from an external service – less than 0.5 s.

The learning results of the developed CNN model are follows:

1. Accuracy of image classification on the training set – 96.6%;

2. Accuracy of image classification on the test set – 65.4%;

3. The average training time for the era – 24 seconds.

The results of time testing for different request:

1. Classification of the image by the trained CNN – 115 ms;

2. Adding a new image to the training set (excluding the execution time of asynchronous processes) – 39 ms;

3. Downloading an image from an external service – 434 ms;

4. Obtaining classification statistics – 5 ms.

Testing results of the application are satisfying the technical requirements.

Conclusion

The model of the CNN and the training dataset have been developed to solve the problem of classification of typical insect pests. A software application for solving the problem of typical insect pests recognition by the transmitted image has been implemented, which allows to automate preventive protection against insect pests in case of integration with IoT-devices on fields. The implemented application satisfies the technical requirements for speed and quality of classification.

Further improvement of the quality of the CNN classification can be done with the help of such methods: GPU calculation acceleration; parallelization of data processing; construction of structurally more complex architectures of the CNN, etc.

References

1. *Dreyfus, G.* (2005) Neural networks: methodology and applications. Springer-Verlag, Berlin. 498 p. https://doi.org/10.1007/3-540-28847-3
2. *Doan, Q. V., Le, T. D., Le, Q. D., & Kang, H.-J.* (2018) A neural network–based synchronized computed torque controller for three degree-of-freedom planar parallel manipulators with uncertainties compensation. International Journal of Advanced Robotic Systems. 15(2). pp. 1-13. https://doi.org/10.1177/1729881418767307
3. *Shymkovych, V., Telenyk, S., Kravets, P.* (2021) Hardware implementation of radial-basis neural networks with Gaussian activation functions on FPGA. Neural Computing and Applications. 33. pp. 9467-9479. https://doi.org/10.1007/s00521-021-05706-3
4. *Melchert, F., Bani, G., Seiffert, U., Biehl, M.* (2020) Adaptive basis functions for prototype-based classification of functional data. Neural Computing and Applications. 32. pp. 18213-18223. https://doi.org/10.1007/s00521-019-04299-2
5. *Goncalves, S., Cortez, P., Moro, S.* (2020) A deep learning classifier for sentence classification in biomedical and computer science abstracts. Neural Computing and Applications. 32. pp. 6793-6807. https://doi.org/10.1007/s00521-019-04334-2
6. *Kravets, P., Shymkovych, V.* (2020) Hardware Implementation Neural Network Controller on FPGA for Stability Ball on the Platform. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing. 938. pp. 247-256. https://doi.org/10.1007/978-3-030-16621-2_23
7. *P., Kravets, V., Nevolko, V., Shymkovych, L., Shymkovych* (2020) Synthesis of High-Speed Neuro-Fuzzy-ControllersBased 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.
8. *Passalis, N., Tefas, A.* (2020) Continuous drone control using deepreinforcement learning for frontal view person shooting. Neural Computing and Applications. 32. pp. 4227-4238. https://doi.org/10.1007/s00521-019-04330-6
9. *Zhao, Z., Zheng, P., Xu, S., Wu, X.* (2019) Object detection with deep learning: a review. IEEE Transactions on Neural Networks and Learning Systems. 30. pp. 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865
10. *Kravets, P.I., Shimkovich, V.N., Ferens, D.A.* (2015) Method and algorithms of implementa-

tion on PLIS the activation function for artificial neuron chains. Elektronnoe Modelirovanie. 37(4). pp. 63-74. (in Russian)
11. *Hong, Q., Li, Y., Wang, X.* (2020) Memristive continuous Hopfield neural network circuit for image restoration. Neural Computing and Applications. 32. pp. 8175-8185. https://doi.org/10.1007/s00521-019-04305-7
12. *V., Shymkovych, V., Samotyy, S., Telenyk, P., Kravets, T., Posvistak* (2018) A real time control system for balancing a ball on a platform with FPGA parallel implementation. Technical Transactions. Poland. 5. pp. 109-117. https://doi.org/10.4467/2353737XCT.18.077.8559
13. *Shao, L., Zhu, F., Li, X.* (2015) Transfer learning for visual categorization: a survey. IEEE Transactions on Neural Networks and Learning Systems. 26. pp. 1019-1034. https://doi.org/10.1109/TNNLS.2014.2330900
14. *P.I., Kravets, T.I., Lukina, V.M., Shymkovych, I.I., Tkach* (2012) Development and research the technology of evaluation neural network models MIMO-objects of control. Visnyk NTUU "KPI" Informatics operation and computer systems. 57. pp. 144-149. (in Ukrainian)
15. *Y., LeCun, B., Boser, J.S., Denker, D., Henderson, R.E., Howard, W., Hubbard, L.D., Jackel* (1989) Backpropagation applied to handwritten zip code recognition. Neural computation. 1. pp. 541-551.
16. *Asifullah Khan, Anabia Sohail, Umme Zahoora, Aqsa Saeed Qureshi* (2020) A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review. 53. pp. 5455-5516. https://doi.org/10.1007/s10462-020-09825-6
17. *Kingma, D.P., Ba, J.* (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
18. *H., Yessou, G., Sumbul, B., Demir* (2020) A Comparative Study of Deep Learning Loss Functions for Multi-Label Remote Sensing Image Classification. IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium. pp. 1349-1352. https://doi.org/10.1109/IGARSS39084.2020.9323583
19. *S., Ioffe, C., Szegedy* (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, PMLR. 37. pp. 448-456
20. *T.N., Mundhenk, B.Y., Chen, G., Friedland* (2020) Efficient Saliency Maps for Explainable AI. European Conference on Computer Vision (ECCV). https://arxiv.org/abs/1911.11293

Received: 17.11.2021

About authors:

Bezliudnyi Yurii Serhiyovych,
5th year student of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".
https://orcid.org/0000-0003-1950-9790

Shymkovych Volodymyr Mykolayovych,
Candidate of Technical Sciences,
Associate Professor of the Department of Information Systems and Technologies of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". Number of scientific publications in Ukrainian publications – more than 30.
Number of scientific publications in foreign publications – more than 10.
Hirsch index – 3.
https://orcid.org/0000-0003-4014-2786

Doroshenko Anatoliy Yukhymovych,
Doctor of Physical and Mathematical Sciences, Professor.
Head of the Department of Computation Theory, Professor of the Department of Information Systems and Technologies of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". Number of scientific publications in Ukrainian publications - more than 190.
Number of scientific publications in foreign publications - more than 80.
Hirsch index - 6.
http://orcid.org/0000-0002-8435-1451

Affiliations:

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute",
37 Peremohy Avenue

Institute of Software Systems of the National Academy of Sciences of Ukraine, 03187,
Kyiv-187, 40 Akademika Glushkova Avenue.
Phone: (044) 526 3559
E-mail:
uraura05052000@gmail.com,
v.shymkovych@kpi.ua,
doroshenkoanatoliy2@gmail.com