ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 6, No. 2, 2021

SOFTWARE IMPLEMENTATION OF THE ALGORITHM FOR RECOGNIZING PROTECTIVE ELEMENTS ON THE FACE

Voloshyn Mykola, Yevhenii Vavruk

Lviv Polytechnic National University, 12, S. Bandery Str., Lviv, 79013, Ukraine.
Authors' e-mail*: Mykola.Voloshyn.mki.2020@lpnu.ua, Yevhenii.Y.Vavruk@lpnu.ua*

https://doi.org/10.23939/acps2021.02.155

Submitted on 29.10.2021

***Abstract*: The quarantine restrictions introduced during COVID-19 are necessary to minimize the spread of coronavirus disease. These measures include a fixed number of people in the room, social distance, wearing protective equipment. These restrictions are achieved by the work of technological control workers and the police. However, people are not ideal creatures, quite often the human factor makes its adjustments. That is why in this work we have developed software for determining the protective elements on the face in real time using the Python scripting language, the open software libraries OpenCV v4.5.4, TensorFlow v2.6.0, Keras v2.6.0 and MobileNetV2 using the camera.**

The training program uses a prepared set of photos from KAGGLE – with a mask and without a mask. This set has been expanded by the authors to include different types of masks and their location. Using TensorFlow, Keras, MobileNetV2, a model is created to study the neural network by analyzing images. The generated neural network uses a model to determine the masks. You can preview the learning result of the network – it is presented as a graphic file. A program that uses the connected camera is then launched and the user can test the operation.

This model can be easily deployed on embedded systems such as Raspberry Pi, Google Coral, and become a hardware and software automated system that can be used in crowded places – airports, shopping malls, stadiums, government agencies and more.

***Index Terms*: Computer Vision, Deep Learning, TensorFlow, OpenCV, Keras, MobileNetV2, KAGGLE**

I. INTRODUCTION

SARS-CoV-2 is a virus that causes the development of respiratory diseases in humans (including acute respiratory disease COVID-19) and is transmitted from person to person [1]. The World Health Organization has described the outbreak of COVID-19 as a pandemic with extremely high population damage across the world [2]. Therefore, governments have developed a number of restrictions to minimize the spread of coronavirus – to wear protective elements on the face, to maintain social distance.

To comply with these requirements, automated subject monitoring systems are being actively developed using various software and hardware systems [3], [4].

Such systems are not perfect because they have a number of disadvantages. Among them: the lack of recognition of different types of protective elements, incorrect identification of the position of protective elements on the face.

This paper presents a software algorithm that minimizes the impact of the above shortcomings and is based on the use of technologies of artificial intelligence [5], deep learning [6], computer vision [7]. The project is implemented using open source software packages: OpenCV [8], TensorFlow [9], [12], Keras [10], MobileNetV2 [11].

This algorithm can be transferred to embedded systems, which allows you to develop a full-fledged automated system for recognizing protective elements on the face.

II. PURPOSE OF THE ARTICLE

The purpose of this article is to develop software for the identification of security features under quarantine restrictions, determining the probability of recognizing protective elements. The software must work in real time. It is necessary to achieve high accuracy in determining the contours of the face, the correct placement and use of different types of protective elements on the face. There is a hope for software deployment capabilities in embedded systems with the addition of hardware. There are also additional goals: creation and methods of learning the neural network, development of the structural algorithm of the program.

III. BASIC APPROACHES TO SOFTWARE CREATION

Coronavirus disease has caused irreparable damage to humanity [1], [2]. The development of an algorithm for detecting protective elements on the face involves the use and combination of many technologies [3], [4]. With the development of information technology in the modern world, an important role is given to computer tools that are able to more accurately and efficiently perform tasks where a person is not able to give good results over a long period of time or a person cannot perform certain tasks. Therefore, there are technologies that can facilitate the implementation of such processes.

Neural networks are used to solve facial recognition problems. An artificial neural network is a

mathematical model modeled after a network of biological nerve cells. The elements of such a network – neurons – the smallest computing units – are a kind of "nano-processors", each of which receives information, performs simple calculations on it and transfers it further. In branched and complex neural networks, neurons are grouped into "layers", and the neuron of each layer can receive information from all neurons of the previous layer, and transmit, respectively, to all neurons of the next one [5].

To recognize the protective elements on the face, a convolutional neural network is used – the main tool for the classification and recognition of objects, faces in photographs, speech recognition (Fig. 1).

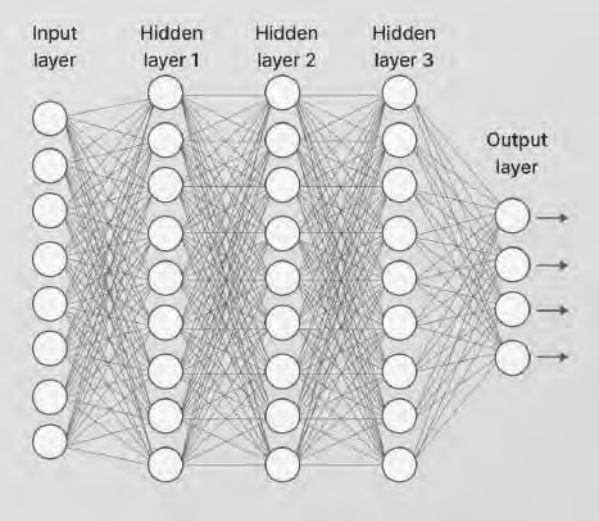

Fig. 1. Convolutional neural network architecture

Machine learning allows computers to learn on their own. This is possible thanks to the processing power of modern computers, which can easily process large datasets.

Supervised learning involves the use of labeled datasets containing inputs and expected outputs. When you train a neural network with supervised learning, you supply both inputs and expected outputs.

If the result generated by the neural network is incorrect, it will adjust its calculations. It is an iterative process that ends when the network stops making mistakes.

An example of a supervised learning problem is image processing. The neural network learns to make a weather forecast using historical data. The training data includes inputs (pressure, humidity, wind speed) and outputs (temperature).

Unsupervised learning is machine learning using datasets with no specific structure.

In the process of training the neural network, it performs a logical classification of data independently. An example of a problem with unsupervised learning is predicting the behavior of online store visitors. In this case, the network is not trained on labeled data. Instead, it classifies the input data on its own and answers the question of which users are most likely to buy different products.

Deep learning is a machine learning technique. Deep learning allows you to train a model to predict the outcome from a set of inputs [6]. Both supervised and unsupervised learning can be used to train the network (Fig. 2).

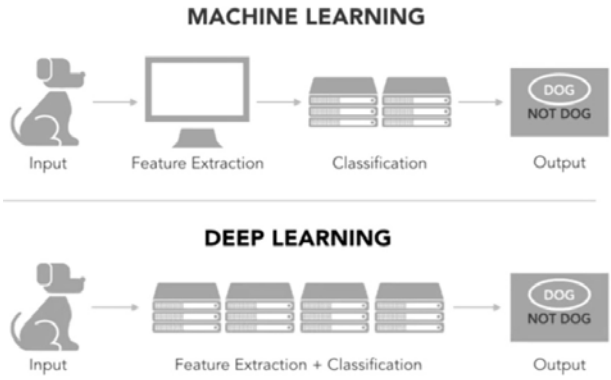

Fig. 2. The difference between deep and machine learning

Deep learning involves the use of a large database of examples – datasets. They are needed to train the neural network that creates the model – a set of data with which the network is able to predict the outcome and make decisions.

Depending on the types of tasks, there are many resources with datasets. The most common are KAGGLE (https://www.kaggle.com/omkargurav/face-mask-dataset), Google Dataset. It is also advisable to use GitHub or create your own database using LabelImg, for example.

For image processing, modern resources provide for the use of another function of artificial intelligence – computer vision.

As a technology discipline, computer vision seeks to apply the theories and models to the creation of computer vision systems. Examples of the application of such systems can be: process control systems (industrial robots, autonomous vehicles), video surveillance systems, systems for organizing information (for example, for indexing image databases), systems for modeling objects or the environment (analysis of medical images, topographic modeling), interaction systems (for example, input devices for a human-machine interaction system), augmented reality systems, computational photography, for example, for mobile devices with cameras [7].

One of the most common software for implementing computer vision is OpenCV, is an open-source library of general-purpose computer vision, image processing and numerical algorithms. Implemented in C/C++, there are also versions for Python, Java, Ruby, Matlab and other languages.

There is a generally accepted scheme for performing the task of image identification using OpenCV (Fig. 3).

Image processing using OpenCV is a complex set of subtasks [8]. However, all these tasks are described by several functions, which makes this library quite flexible and often used even by large IT businesses.

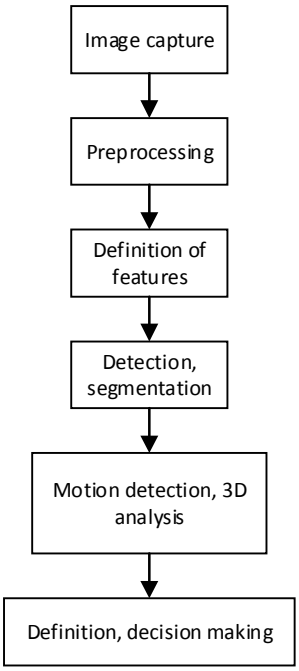

Fig. 3. General block diagram of an OpenCV application

IV. DESCRIPTION OF COMPONENTS AND MODE OF OPERATION OF THE ALGORITHM

To develop an algorithm for identifying protective elements on the face, the Python scripting language was used, which has the support of libraries for the creation and use of neural networks with appropriate auxiliary resources.

TensorFlow and Keras v2.6.0 were used to create the convolutional neural network. These are open program libraries with the functions of creating neural networks and their training – machine and deep learning. They are complementary – using TensorFlow to create and configure neural networks [9], and Keras – a set of application programming interfaces used for network training [10].

Because in-depth network training depends on the amount of data, this process can be quite time consuming. To optimize this process, additional software components are used – MobileNetV2 [11]. This is a lightweight deep neural network for embedded and mobile devices. The network is based on the use of deep convolution – processing (convolution) occurs for each input channel separately (input channel – the division of the input flow of information into convolutional cores). The network uses 32 convolutional filters and has 3 processing layers: point convolution – expansion layer (creates a reflection of the input stream in a large dimension), depthwise convolution and the final convolution 1x1 – bottleneck layer. This allows you to achieve the speed of processing input data and easily transfer the network between platforms.

A neural network training requires a dataset, a large set of data that allows a neural network to make predictions and identify objects. KAGGLE image dataset is used for this purpose.

OpenCV, which implements computer vision, is used to run the program in real time. It is necessary to determine the contours of the user's face and subsequent identification of security features.

The structure of software algorithm involves 2 stages: neural network training and detection (Fig. 4).

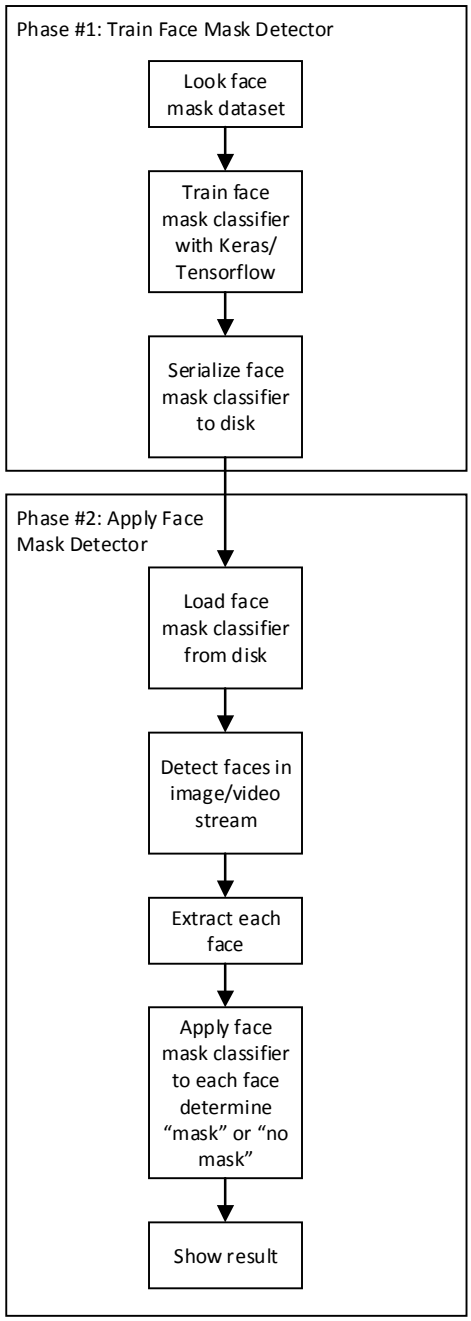

Fig. 4. Basic block diagram of the algorithm

The dataset has been modified to determine the different types of masks and their correct location. It contains two folders: “with mask” (2150 images) and “without mask” (2248 images). Each image is loaded with a photo size transformation to determine certain features (Fig. 5).

At the training stage, the speed, number of packets and the amount of data in each packet are indicated.

For training, a full image is provided, measuring approximately 224x224 pixels in high definition – for proper identification of faces and masks. This involves taking into account the filling of the photo with colors, contrasts, clear contours of objects in the photo.

Fig. 5. Example of image for processing

The next step is to download and pre-process the data. Pre-processing is the replacement of image size by 224x224 pixels, conversion of images into an array of data in the format [-1,1] – label generation [12].

The next step is to configure the MobileNetV2 network, which involves configuring the main and auxiliary layers of input processing.

Then there is a one-time encoding of the created labels with indexes. These indices will form the future model for which the neural network will work.

To improve the efficiency of the algorithm, the images are also processed directly during training – changing the zoom, the offset parameters, the angle of rotation. This is done using an image generator (ImageDataGenerator) and is additional data when learning the network.

Now, as the architecture is not configured, the data is loaded, processed and recoded into appropriate labels, the process of learning the neural network and generating the model takes place. After that it is necessary to check the selected model on the test data set.

To test the effectiveness of network learning, a pre-test schedule is built on the basis of the MatPlotLib library, which is supported by the Python language. This will allow you to check the correctness of the definition of protective elements on the face in percentage.

If the learning outcomes are satisfactory, the transition to the real-time recognition program is performed.

After connecting OpenCV v4.5.4 parameters are set for library work: image (input, with borders for the conclusion), face (catalog for definition of a contour), model (created during training), confidence (for filtering of weak detection of faces – value of 50% will be enough).

The next step is to load the generated model and pre-process to capture the frame size of the future display and increase, done by replacing the previous image by increasing the size to 300x300 pixels and calculating the average value.

The value determines the face to localize all faces in the images. The result is checked with a threshold of confidence – they must be appropriate [13].

A boundary triangle is created to identify the face, by linking the results obtained earlier and the confidence threshold. An additional step to correctly identify the face in this test is the model that was created during the training of the system. Next, determine the possible position of the mask on the face.

The last step is to set the label class and display the result to the user. The label class is determined based on the probabilities obtained when processing the network model and the purpose of this label of the corresponding color: green – the user in the mask, red – the user without the mask. The text of the label with the name of the class and the probability of detection of the security element and the rectangle of the front frame is formed using the OpenCV drawing functions. The processed image is shown on the display.

V. VERIFICATION

In order to check the algorithm, it must be tested.

The proposed algorithm uses a Lenovo IdeaPad Gaming I15 laptop with a six-core Intel Core i7-9750 processor, frequency of 2.6 GHz. The size of RAM is 16 GB. Operation system is Windows 10 Pro.

The neural network was trained on a data set that provides 4398 images. They contain photos of faces without protective elements and with them (also involves the use of different types of masks).

Learning takes place by epochs – stages of learning that the network goes through. Number of epochs – 20. If the result is not high enough for higher efficiency, the number of epochs can be increased. Each epoch is divided by the amount of data loaded – this is a constant that divides the input array into equal intervals. This work uses 2 intervals (Fig. 6). On a computer with the configuration described above, according to the available dataset from KAGGLE, neural network training takes 53 minutes.

```
Epoch 1/20
1759/1759 [==============================]
Epoch 2/20
1759/1759 [==============================]
Epoch 3/20
1759/1759 [==============================]
Epoch 4/20
1759/1759 [==============================]
Epoch 5/20
1759/1759 [==============================]
Epoch 6/20
1759/1759 [==============================]
```

Fig. 6. Stages of deep learning

After training and creating a model of work, MatPlotLib v3.4.3 generates a graph with the percentage of accuracy and loss. The result showed that the accuracy of masking detection in this network at the end of training is approximately 99%, which is a good indicator and allows us to test our model in real time (Fig. 7).

The next step in testing is to run a real-time identification algorithm.

Since this is a software algorithm, it does not involve the use of additional hardware resources, it is enough to use a local camera built into any modern laptop. Video streaming through the camera is done by activating it using the VideoStream function. The result of the camera – identification of the user, determining the contour of his face and determining the status of the user – with or without a mask (Fig. 8).

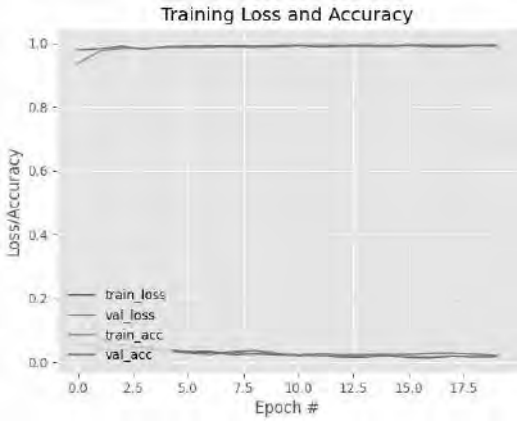

Fig. 7. Graph of calculation of accuracy and loss in deep learning and model generation

Fig. 8. Identification of the user's face without a mask

The real-time face recognition time in a video stream is approximately 200 milliseconds. Depending on the location of the head, the probability of face identification ranges from 95 to 100 percent.

The second stage of verification is the identification of the usual protective mask on the face and verification of the result. (Fig. 9).

The real-time face recognition time in a video stream is approximately 300 milliseconds. Depending on the location of the head, the probability of facial identification ranges from 95 to 100 percent.

The next step is to check the identification of protective elements in the wrong position of the mask – unclosed nose (Fig. 10). The class identification time is 320 milliseconds. The probability of recognition is in the range of 86–97 percent.

Fig. 9. Identification of the user's face with a mask

Fig. 10. Check the identification with the nose not closed

Identification of other types of masks is also checked. (Fig. 11). Recognition time is 300 milliseconds. The probability of class recognition is 95-100 percent.

Fig. 11. Verification of identification of homemade protective masks

The results of the study showed that this software product defines the contours of the face, displays them in the video stream and identifies the protective elements. It displays the class to which the user belongs and the probability of the recognition. The software also identifies protective elements of different types. The program is able to determine whether the contour of the mask on the face is in the correct position – if the mask does not cover the nose, the user is identified in the class “NoMask”.

VI. CONCLUSION

In this paper, an algorithm for determining the protective elements on the face in quarantine measures was developed and tested. The algorithm of work was created and described. The project is implemented in Python scripting language using computer vision. The neural network is taught by the method of deep learning using open data sets and libraries. The results showed that the recognition is performed with accuracy: 95–100 percent for the face, 95–100 percent for the recognition of the mask, 86–97 percent when the mask is in the wrong position, 95100 percent for the identification of other types of masks. The resources used in this work allow you to deploy software on embedded systems, such as Raspberry Pi, which will further improve the hardware features to create a software-hardware automated system.

REFERENCES

[1] World Health Organization. Transmission of SARS-CoV-2 – implications for infection prevention precautions: Scientific brief. July, 2020. Available at: https://apps.who.int/iris/bitstream/handle/10665/333114/WHO-2019-nCoV-Sci_Brief-Transmission_modes-2020.3-eng.pdf (Accessed: 18 November 2021).

[2] D. M. Morens, Gregory K. Folkers, and Anthony S. Fauci. What is a Pandemic? August, 2009. Available at: https://academic.oup.com/jid/article/200/7/1018/903237 (Accessed: 18 November 2021).

[3] Henderi, A. Setiani Rafika, H. L. Hendric Spits Warnar, M. A. Saputra. An Application of Mask Detector For Prevent Covid-19 in Public Services Area. Henderi et al 2020 J. Phys.: Conf. Ser. 1641 012063. doi:10.1088/1742-6596/1641/1/012063

[4] G. K. Jakir Hussain, R. Priya, S Rajarajeswari, P. Prasanth, N. Niyazuddeen. The Face Mask Detection Technology for Image Analysis in the Covid19 Surveillance System. G K Jakir Hussain et al 2021 J. Phys.: Conf. Ser. 1916 012084. doi:10.1088/1742-6596/1916/1/012084.

[5] S. J. Russel, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Upper Saddle River, New Jersey 07458, 2010, pp. 727–737. ISBN-13: 978-0-13-604259-4. Available at: https://cs.calvin.edu/courses/cs/344/kvlinden/resources/AIMA-3rd-edition.pdf (Accessed: 18 November 2021).

[6] C. Ranjan, Understanding Deep Learning Application in Rare Event Prediction, 1st ed., USA, 2020, pp. 13–15. ISBN: 9798586189561. Available at: https://www.researchgate.net/publication/348077077_Understanding_Deep_Learning_Application_in_Rare_Event_Prediction (Accessed: 18 November 2021).

[7] G. Kudrayvtsev, Fundamentals of Computer Vision, May, 2020, pp.14-16. Available at: https://drive.google.com/file/d/11CoPBCQHwVTlv7_u1UKSdo3xo1HlCncj/view (Accessed: 18 November 2021).

[8] J. Minichino, J. Howse, Learning OpenCV 3 Computer Vision with Python, 2nd ed., Birmingham, 2015, pp. 209–228. ISBN 978-1-78528-384-0. Available at: https://repository.unikom.ac.id/67052/ (Accessed: 18 November 2021).

[9] T. Hope, Ye. S. Resheff, I. Lieder, Learning TensorFlow: A Guide to Building Deep Learning Systems, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, pp. 23–40. ISBN: 978-1-491-97851-1. Available at: https://www.academia.edu/40118139/TensorFlow_A_GUIDE_TO_BUILDING_DEEP_LEARNING_SYSTEMS (Accessed: 18 November 2021).

[10] A. Gulli, S. Pal, Deep Learning with Keras: Implemented neural networks with Keras on Theano and TensorFlow, Birmingham, 2017, pp. 88–100. ISBN: 978-1-78712-842-2. Available at: https://sites.google.com/site/9520camilemoh3/1oidarkAbetuul7uJhyA911 (Accessed: 18 November 2021).

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[12] R. Zhang, TensorFlow 2 Tutorial, 2020, pp. 4–12. Available at: https://itbook.store/books/1001606140961 (Accessed: 18 November 2021).

[13] G. Garrido, P. Joshi, OpenCV 3.x with Python By Example, 2nd ed., Birmingham, January, 2018. ISBN: 978-1-78839-690-5. Available at: https://libribook.com/view1/9744 (Accessed: 18 November 2021).

Mykola Voloshyn is a second-year Master's student of the Department of Electronic Computers at Lviv Polytechnic National University. In 2020, he received a bachelor's degree in Computer Engineering, Department of Specialized Computer Systems. Developed automated access control and management systems for organizing work in various fields.

Areas of interest include embedded systems engineering, such as Robotics, application development for processor modules (such as STM32, PSoC4, PSoC6), Automated Testing, the Internet of Things, Artificial Intelligence. He is interested in designing and modeling modules using VHDL and Verilog hardware description languages.

Yevhenii Ya. Vavruk is an associate professor of the Department of Electronic Computers at Lviv Polytechnic National University. In 1975 he graduated from the Department of Electronic Computers of Lviv Polytechnic Institute with a degree in Electrical Engineering. During 1975-2000 he worked at Lviv Research Radio Engineering Institute, held the position of head of the group for the development of processors and signal processing systems. He was engaged in the design, commissioning, testing of hardware channels for the exchange of information and measuring systems, processors for the control of moving objects, high-performance systems and specialized processors of radar systems.

The topic of the candidate's dissertation is "Functionally-oriented processors for onboard control and information processing systems". From 2001 to 2005 he held the position of senior lecturer at the Department of Electronic Computers. Since 2005 he has been an associate professor.

Research interests – digital signal and image processing (design of algorithms and hardware for digital signal and image processing), development of parallel computational algorithms and structures.

He is the author of 98 copyright certificates for inventions in the field of computer technology (32 introduced into production), 33 articles in professional journals, 28 conference abstracts, 27 educational and methodological developments (including 7 textbooks).