Diachenko Liliia, Lazoriak Oleksandr, Dobrovolsky Yurii, Prokhorov Georgii, Shumyliak Liliia. Neural networks for the Recognition of X-ray Images of Ailments for Covid-19. Journal of Education, Health and Sport. 2022;12(6):26-38. eISSN 2391-8306.
DOI http://dx.doi.org/10.12775/JEHS.2022.12.06.002
https://apcz.umk.pl/JEHS/article/view/JEHS.2022.12.06.002
https://zenodo.org/record/6490613

The journal has had 40 points in Ministry of Education and Science of Poland parametric evaluation. Annex to the announcement of the Minister of Education and Science of December 21, 2021. No. 32343. Has a Journal's Unique Identifier: 201159. Scientific disciplines assigned: Physical Culture Sciences (Field of Medical sciences and health sciences); Health Sciences (Field of Medical Sciences and Health Sciences).

Punkty Ministerialne z 2019 - aktualny rok 40 punktów. Załącznik do komunikatu Ministra Edukacji i Nauki z dnia 21 grudnia 2021 r. Lp. 32343. Posiada Unikatowy Identyfikator Czasopisma: 201159. Przypisane dyscypliny naukowe: Nauki o kulturze fizycznej (Dziedzina nauk medycznych i nauk o zdrowiu); Nauki o zdrowiu (Dziedzina nauk medycznych i nauk o zdrowiu).

Received: 03.04.2022. Revised: 20.04.2022. Accepted: 26.04.2022.

Neural networks for the Recognition of X-ray Images of Ailments for Covid-19

Liliia Diachenko[a], Oleksandr Lazoriak[a], Yurii Dobrovolsky[a], Georgii Prokhorov[a], Liliia Shumyliak[a]

[a] *Yuriy Fedkovych Chernivtsi National University, Kotsyubynsky 2, Chernivtsi, 58012, Ukraine*

l.dyachenko@chnu.edu.ua (Liliia Diachenko);
oleksandr.lazoriak@chnu.edu.ua (Oleksandr Lazoriak);
y.dobrovolsky@chnu.edu.ua (Yurii Dobrovolskyi),
g.prokhorov@chnu.edu.ua (Georgii Prokhorov),
l.shumylyak@chnu.edu.ua (Liliia Shumyliak)

Abstract
This investigation analyzes the current state of neural networks, considers the available types, optimizers used for training, describes their benefits and disadvantages. The task of computer vision is defined and the answer to the question why the use of neural networks is an important task today is given. The powerful neural network from Google was proposed as an example and its algorithm is described in detail. Studies have shown how to configure models to get high performance.

Keywords: Convolutional neural network, computer vision, optimizers, X-rays, model, Covid-19, machine learning.

1. Introduction

At the moment, the task of recognizing X-rays images of patients with Covid-19 is extremely important. Proper recognition of indicators of the disease in the early stages, can significantly increase the chances of rapid recovery of a patient. Of course, the recognition of images can be done manually by highly qualified doctors, but in periods of pandemia , such staff is usually not enough, and in some medical institutions (especially in small towns) the amount of such a specialists in principle is limited. Therefore, automating the recognition of such images is of a high need.

One of the ways of automations of recognizing such images is n Artificial Neural Networks, which is used for computer vision tasks [1]. Recognition of objects in images and their classification is a task that significantly reduces the cost of various resources (time, money, etc.). An example is DeepFake [2,3], which allows you to replace people's faces with video and is distributed as an open source.

2. Review of neural network development tools

To create an efficient and modern neural network, it is important to use modern methods and tools of development. The necessary tools are: a programming language, a virtual environment that would allow you to easily run the application on another personal computer and a framework for working with neural networks.

The most commonly used programming language among scientists and engineers is Python [4]. It has several advantages over competitors in the paradigm of scientific work. First of all, it is the speed of mastery and strong support of the community. Many Python libraries for AI and ML, which significantly reduce costs and speed up development. Simple syntax and readability help test complex processes quickly and make the language understandable to everyone.

As you can see, Python has many advantages, but there are some limitations: because it is an interpreted language, Python can be slower than other compiled languages; Matlab lacks Python counterparts for several toolkits; MatPlotLib also has some problems, including the lack of a unified interfaces.

Popular frameworks are Pandas [5], Numpy [6], Scipy [7]. The list of libraries for working with neural networks includes PyTorch [8], Theano [9], Caffe [10], TensorFlow [11] and others.

There are several technologies in the development of artificial intelligence.

Apache MxNet [12] is a fast and flexible open source deep learning framework that supports state-of-the-art neural network technology, including convolutional neural networks and long-term short-term memory. Compared to TensorFlow, MXNet has less open source community, bug fixes and other features take longer due to a lack of basic community support. Although MxNet is widely used by many technical organizations, MxNet is not as popular as Tensorflow.

PyTorch [13] is an open source Python machine learning library based on the Torch machine learning library and well optimized for graphics processing (GPU). Deep neural networks and tensor calculations with acceleration on the graphics card are key features of the library. Disadvantages include the lack of interfaces for monitoring and visualization, such as TensorFlow. Also, PyTorch is a new deep learning system that currently has less community support.

Theano [14] is a Python library that allows you to define, optimize, and evaluate mathematical expressions using multidimensional arrays. The main purpose of the library is to work quickly with large neural networks. The disadvantage is that it has a complex syntax compared to Tensorflow, so it will not be the best choice for beginners.

Keras [15] is one of the main open source libraries for neural networks and machine learning. The feature is that it can work with Tensorflow, Deeplearning4j, MXNet, Microsoft Cognitive Toolkit (CNTK), Theano. Also, it implements optimizers, neural layers, layer activation functions, initialization schemes, cost functions and control models. Keras is not a full-fledged ML full library, but instead extends the functionality of other libraries.

Tensorflow [16] is an open library for machine learning. Provides an API for working with programming languages such as Java, C ++, Haskell, Go and Python. In this library you can build neural networks for speech recognition, highlighting faces in photos, determining

the similarity of images and more. It has an excellent architecture that allows it to run on phones, servers and desktops. The main advantage is abstractions, which allow you to focus on the logic of the application, rather than on the small details of the implementation of certain algorithms. Disadvantages include the fact that TensorFlow is a bit slow compared to frameworks such as MxNet and CNTK, debugging can be a daunting task and a lack of OpenCL support.

To solve this problem, it was first decided to use existing open source models. Our model is based on the DenseNet model [17]. The advantages include the fact that a tight connection achieves fewer parameters and higher accuracy compared to ResNet [18] and Res - Pre-Activation ResNet [19].

In DenseNet, each layer receives additional inputs from all previous layers and passes its own function maps to all subsequent layers. Concatenation is used. Each layer receives "collective knowledge" from all previous layers.

Because each layer receives function maps from all previous layers, the network can be thinner and more compact, ie the number of channels can be smaller. Growth rate k is the additional number of channels for each layer.

For each layer of the composition apply the packet rate of pre-activation (BN) and ReLU, then 3 × 3 convolution. This is an idea from Pre-Activation ResNet [20].

The monk_v1 framework was used to quickly implement a convolutional neural network. This library implements an add-on to Tensorflow [21] and Keras [22] and contains popular open source models. DenseNet described above was selected from the list. As a result, an accuracy of 45.45% was obtained when passing the test data. When classifying images using this neural network, which is based on the DenseNet model, the results shown in Figures 1, 2, 3 were obtained.

```
Prediction
    Image name:         /content/dataset/Covid19-dataset/test/Normal/0110.jpeg
    Predicted class:      Viral Pneumonia
    Predicted score:      0.5192561149597168
```

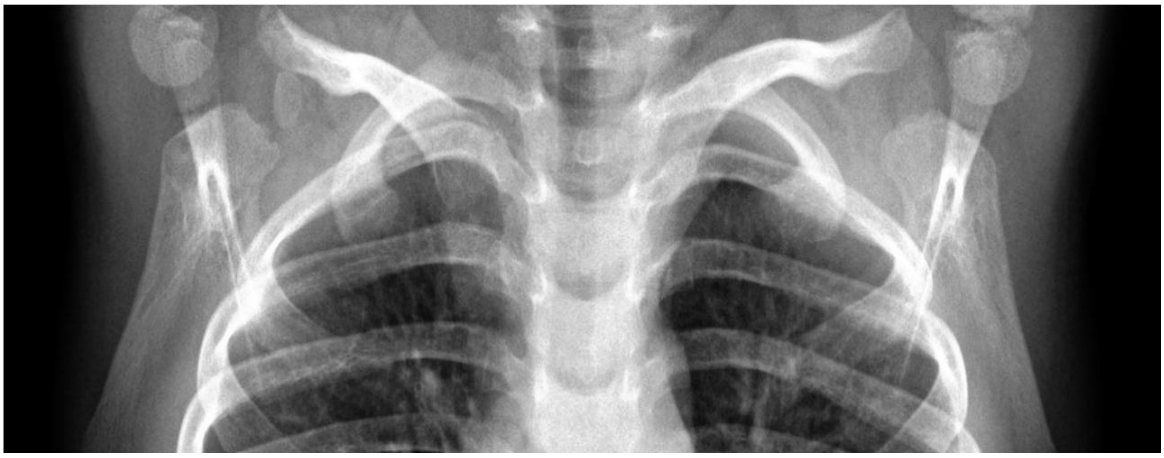

Figure 1: An example of the X-ray classification of a healthy person

```
Prediction
    Image name:         /content/dataset/Covid19-dataset/test/Viral Pneumonia/0104.jpeg
    Predicted class:      Normal
    Predicted score:      0.826201319694519
```

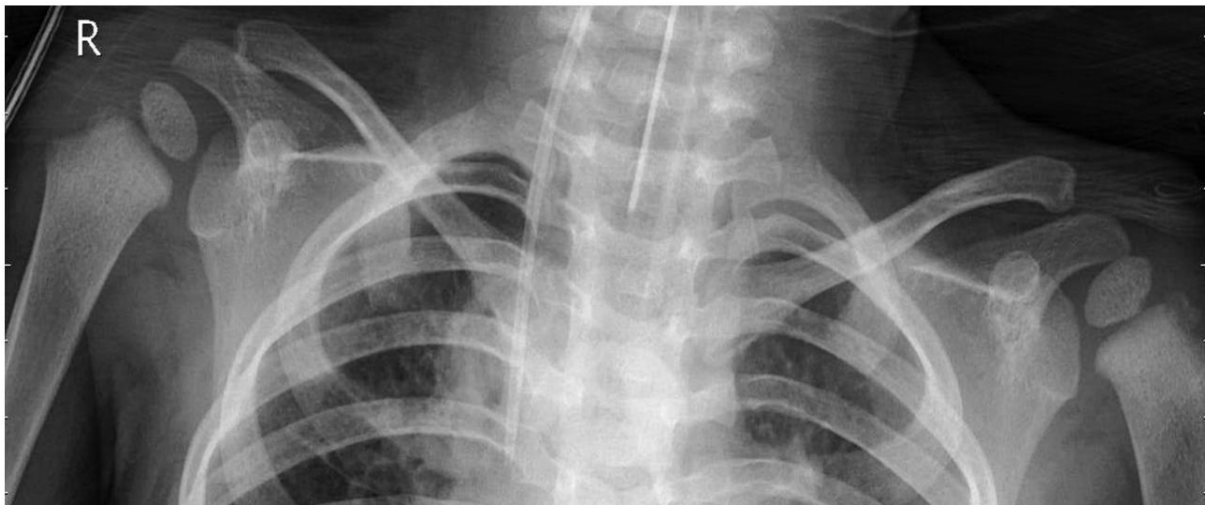

Figure 2: An example of the classification of X-rays of a sick person for viral pneumonia

```
Prediction
    Image name:         /content/dataset/Covid19-dataset/test/Covid/0102.jpeg
    Predicted class:      Covid
    Predicted score:      0.6606042981147766
```

Figure 3: An example of the classification of X-rays of a sick person on coronavirus-19

As you can see in the images, this model could not learn well on the dataset and makes mistakes. Although the patient's picture could be classified correctly, but today the accuracy of 66% is very low.

The problem with this solution is the scale of coverage of possible categories of images with which the used model can work. Therefore, for specific tasks, it is better to use your own models.

That is why it was decided to develop our own model that would increase the accuracy of image recognition to an acceptable level.

3. Implementation and description of the software applcation

3.1. Architecture of a proposed model for a neural network

The list of layers included in the convolutional neural network includes: input layer, 2D convolution layers (Conv2D), aggregation layers for two-dimensional inputs (MaxPolling2D), layer of the average spatial data aggregation operation (AveragePo), input smoother (Flatten), ordinary tightly connected layers (Dense). The hierarchy of layers in the model is presented

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
average_pooling2d (AveragePo (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
-----------------------------------------------------------------
```

in Figure 4.

Figure 4: Representation of the hierarchy of model layers

Figure 5 shows the total number of parameters that can be learned and not.

```
=================================================================
Total params: 14,747,650
Trainable params: 32,962
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 5: Representation of the number of parameters found by the model based on the input data

As can be seen from Figure 5 of the 14 million parameters found by the model, only almost 33,000 are suitable for training.

3.2. Data set in use

Neural network learning is impossible without data, so one of the important tasks is to build a quality dataset.

There are the following sites that contain an open database of open licenses. These include CrowdANALYTIX, DrivenData, crowdAI and Kaggle.

CrowdANALYTIX, DrivenData, crowdAI are platforms for data forecasting competitions. Data is accessed after the user joins the competition.

Kaggle is a platform for date scientists, which competes in forecasting and data processing solutions. Any registered user can download a data set, and other users will offer their own solutions for the data set. The most important reason for choosing is that the data is available without joining the competition, as is necessary in others. That's why Keggle allows you to download data and build your own large datasets from multiple datasets. That is why the Kaggle platform was chosen.

The dataset used was developed according to the following scheme: the first part is data and images of healthy chest images of people, which are designed to teach and verify the original results, the second one - consists of images of sick people on covid-19, annotations in format json and image metadata. The structure of the created dataset is shown in Figure 6.

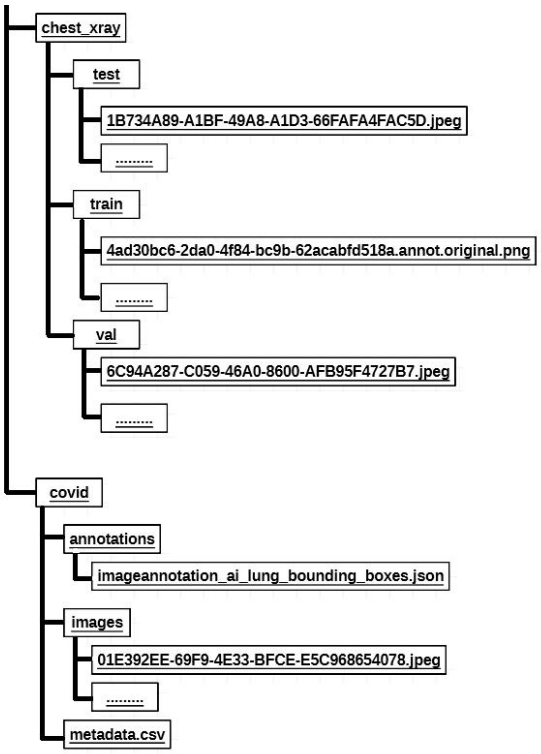

Figure 6: File structue of the used dataset.

3.3. Data flow diagram of the developed software application

Figure 7 shows a data flow diagram of the created software product, from which we can see the sequence of processing and manipulation of data, starting with loading images that need to be recognized and ending with the output of recognition results with forecast.

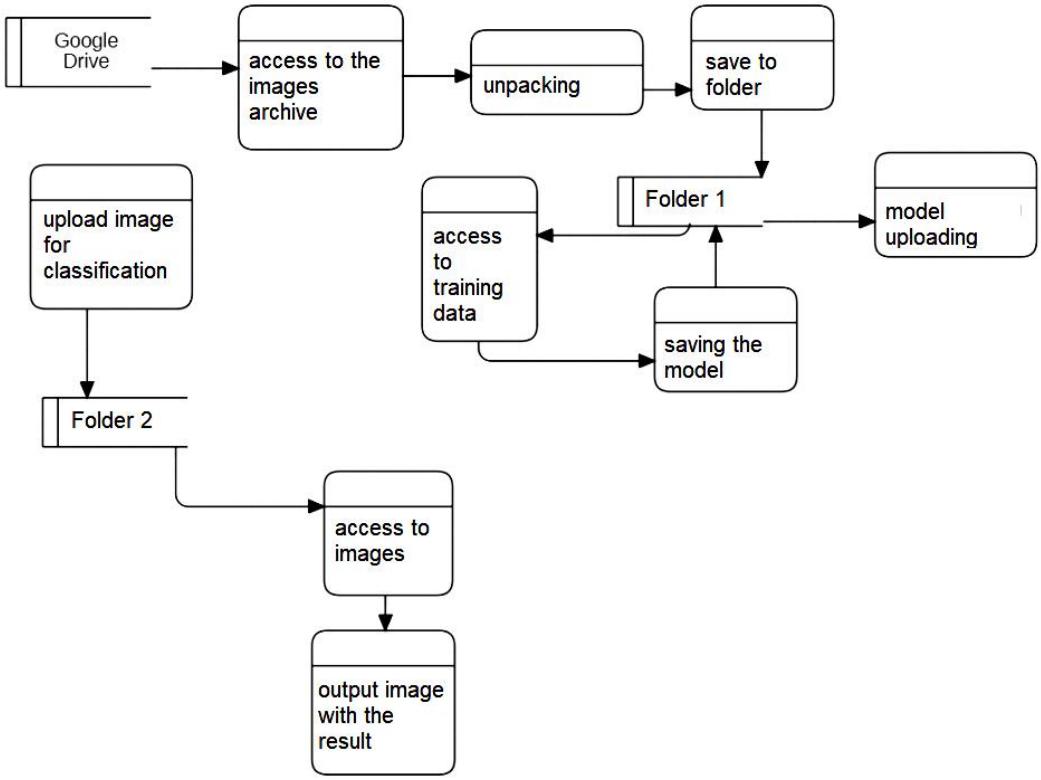

Figure 7: Data flow diagram of the developed software.

3.4. Application components and interfaces

The software product consists of:

- Script for data representation
- implementation of Artificial neural network (ANN);
- implementation of access to the trained model and output of results.

A squeak was used to clear some of the data, which allowed for better output and reduced noise. In the module where the ANN is implemented, the model is created, trained with representative data after script cleaning and saving the model for further use and API architecture. In the module where the model is accessed, it is downloaded and activated for further use. The output of forecasts is also in this module.

In the first module there is work with data, their last preparation for training. Figure 8 shows the images that were prepared and displayed as an array using the matplotlib library.

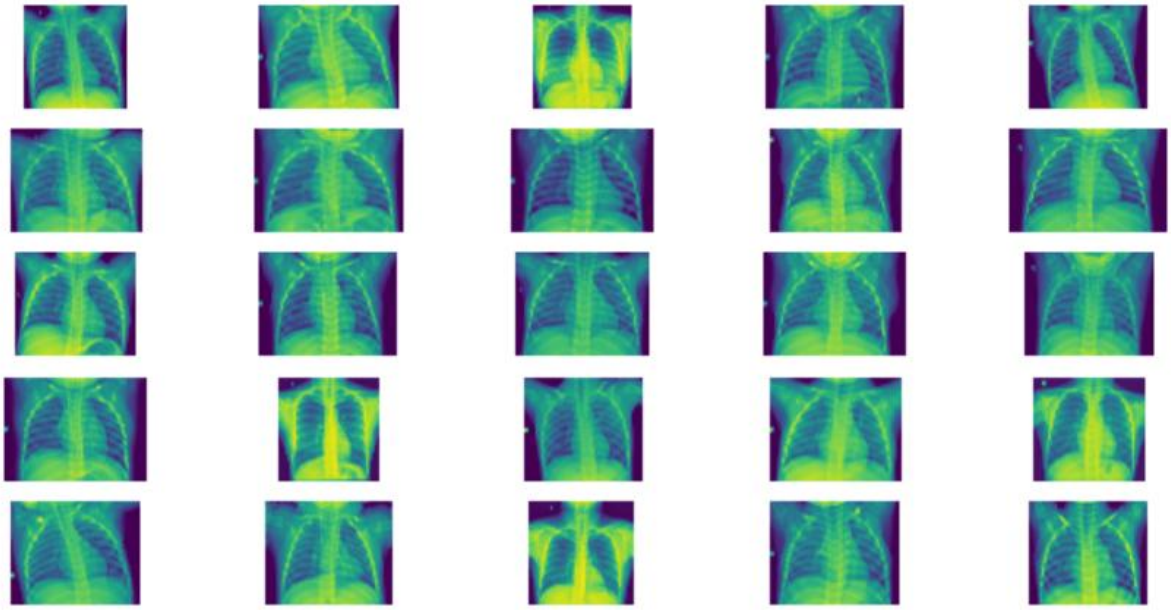

Figure 8: An array of prepared data of chest images of healthy people

Figure 9 shows the last preparation for learning these chest images of sick people in the form of an array.

Figure 9: Array of chest images of patients with Covid-19

After the learning process, a general graph is displayed on the prepared data (see Fig. 10) to understand the effectiveness of the created model and training data using the matplotlib library. The graph shows the main criteria for evaluating the model.

The x-axis shows the percentage scale, where 0 corresponds to 0%, and 1 corresponds to 100%, and the y-axis shows the number of epochs (cycles) of training. As can be seen from the graph, the metrics accuracy of verification and accuracy go to 1, which is a good result for the model and means high accuracy of image classification, which was the goal. And the parameter of loss (error) goes to the minimum values during all epochs of training.

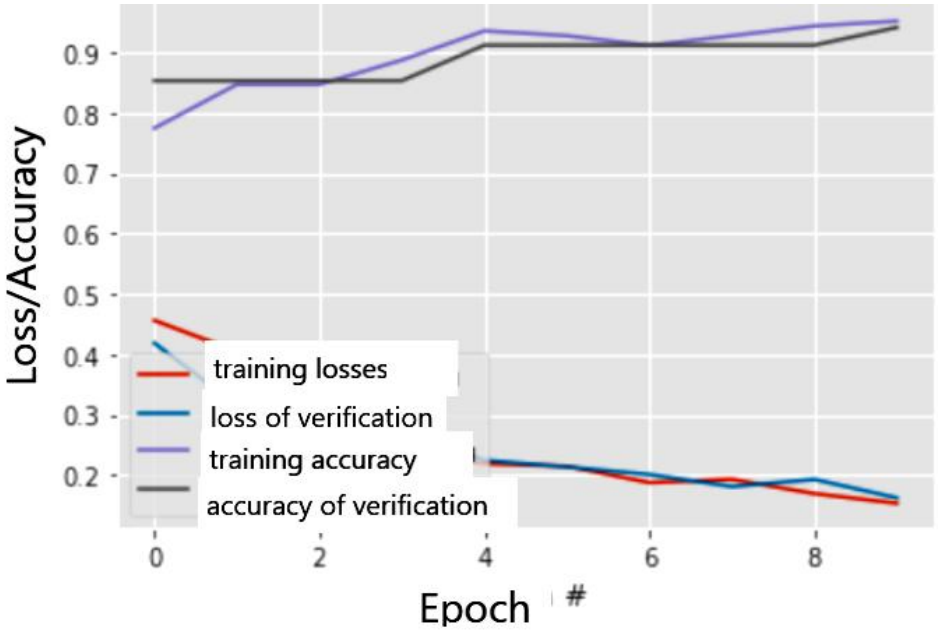

Figure 10: Graph of losses and accuracy of the model

In the third module there is a connection of the created model for the analysis of input images and assignment of their class. The image (see Fig. 11) which shows the tested attempt is a picture of the diseased chest and, as can be seen from the result, the neural network performed the task correct.

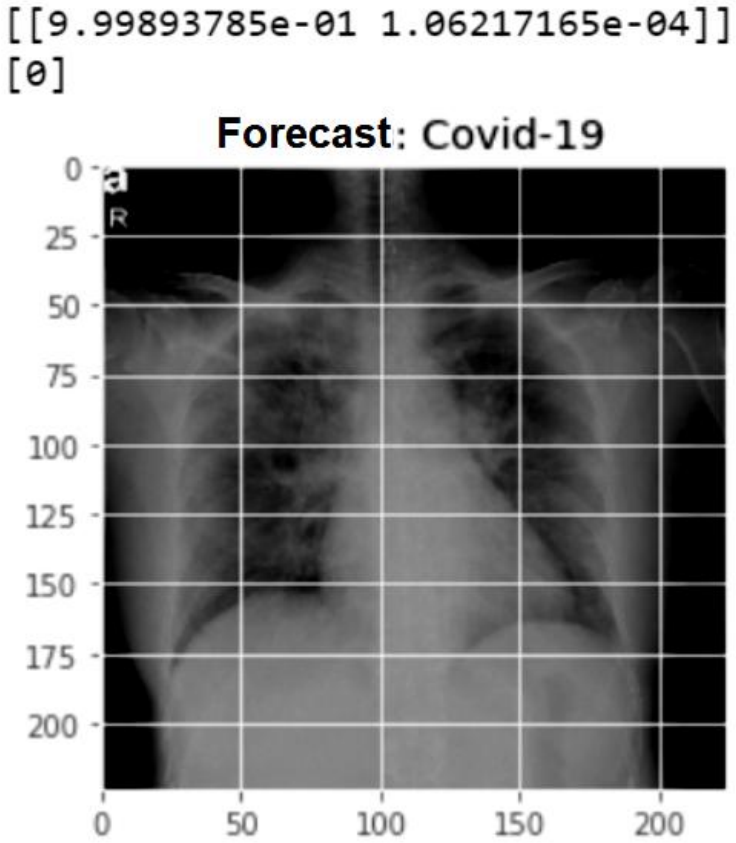

Figure 11: Display the result of the image check

Healthy chest image recognition will also be tested. The neural network also performs this task correctly (see Fig. 12).

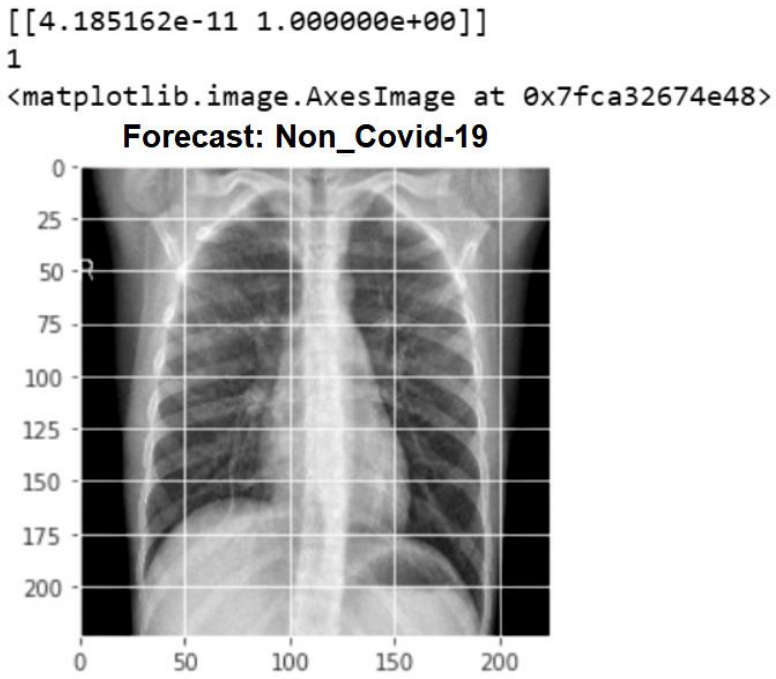

Figure 12: Display the result of the image check

A number of iterations of the test were performed (12 times the chest images alternated in any form), the result was reliable answers during all iterations. That is, using our own model and optimized dataset, we have increased the accuracy of X-ray recognition by more than 30%.

All the processes that take place during the creation of the application were run on a powerful remote virtual machine, which reduced the learning time of ANN. On a virtual machine, the speed of one epoch (see Figure 13) took from 6 seconds 402 milliseconds to 3 seconds 160 milliseconds.

```
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
16/16 [==============================] - 6s 402ms/step
Epoch 2/10
16/16 [==============================] - 3s 161ms/step
Epoch 3/10
16/16 [==============================] - 3s 160ms/step
Epoch 4/10
16/16 [==============================] - 3s 161ms/step
Epoch 5/10
16/16 [==============================] - 3s 163ms/step
Epoch 6/10
16/16 [==============================] - 3s 164ms/step
Epoch 7/10
16/16 [==============================] - 3s 166ms/step
Epoch 8/10
16/16 [==============================] - 3s 167ms/step
Epoch 9/10
16/16 [==============================] - 3s 164ms/step
Epoch 10/10
16/16 [==============================] - 3s 169ms/step
```

Figure 13: Display the speed of training in seconds for each era

4. Conclusion

This investigation analyzes the current state of neural networks, identifies the tasks of computer vision and answers why the use of neural networks is an important task today. Also, the available types of neural networks were considered, the optimizers used for training described their advantages and disadvantages.

Research has shown how to configure models to get high performance in the end, and this has influenced the requirements for your own model.

As an example of the use of open models for neural networks, an open model SNM was created and it was demonstrated why their use is not suitable for specific tasks (recognition accuracy was within 66%).

A functioning convolutional neural network was developed for the classification of images of chest images, which allowed to increase the recognition accuracy compared to open models by more than 30%. This allows us to conclude that the model architecture used and the optimizer are the most optimal for this task. Iterative testing demonstrated the effectiveness of the neural network. Also, the evaluation of the model could be seen on the accuracy coefficients and on the loss functions (accuracy coefficients go to 1, and the loss (error) parameter goes to the minimum values during all epochs of training).

This allows us to conclude that the created ANN can be successfully used for X-ray recognition of patients, providing recognition accuracy of up to 100%

References

[1] Ballard Will. Hands-On Deep Learning for Images with TensorFlow: Packt Publishing Ltd., Birmingham, 2018.

[2] Aldwairi, M., Alwahedi A., Detecting Fake News in Social Media Networks. Procedia Computer Science, 141, 2018, pp. 215–222. doi: 10.1016/j.procs.2018.10.171.

[3] Fletcher, J., Deepfakes, Artificial Intelligence, and Some Kind of Dystopia: The New Faces of Online Post-Fact Performance. Theatre Journal, 70(4), 2018, pp. 455–471. doi:10.1353/tj.2018.0097.

[4] Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., et al., Theano: a Python framework for fast computation of mathematical expressions, 2016. URL: https://arxiv.org/abs/1605.02688.

[5] Daniel Y. Chen., Pandas for Everyone: Python Data Analysis. Addison-Wesley Professional, Boston, 2017.

[6] Mahesh Ravishankar and Vinod Grover, Automatic acceleration of Numpy applications on GPUs and multicore CPUs. CoRR, 2019. URL: http://arxiv.org/abs/1901.03771.

[7] Virtanen, P., Gommers, R., Oliphant, T.E. et al., SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat Methods 17, 2020, pp. 261–272. URL: https://doi.org/10.1038/s41592-019-0686-2.

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, Automatic differentiation in pytorch, in: NIPS Workshop, Long Beach, CA, 2017.

[9] Theano Development Team, Theano: A Python framework for fast computation of mathematical expressions, 2017. URL: https://arxiv.org/abs/1605.02688.

[10] Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, Mohak Shah, Comparative study of caffe, neon, theano, and torch for deep learning, Workshop track – ICLR 2016, San Juan, Puerto Rico, 2016.

[11] L. Yuan, Z. Qu, Y. Zhao, H. Zhang and Q. Nian, A convolutional neural network based on TensorFlow for face recognition, IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017, pp. 525-529. doi: 10.1109/IAEAC.2017.8054070.

[12] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang, MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems, ML Systems workshop at NeurIPS, Montréal, Canada, 2015.

[13] The PyTorch team, Torch Script. URL: https://pytorch.org/docs/stable/jit.html.

[14] Partho Sen, Santosh Lamichhane, Vivek B Mathema, Aidan McGlinchey, Alex M Dickens, Sakda Khoomrung, Matej Orešič, Deep learning meets metabolomics: a methodological perspective, Briefings in Bioinformatics, Volume 22, Issue 2, March 2021, pp. 1531–1542. URL: https://doi.org/10.1093/bib/bbaa204

[15] Taylor B Arnold, KerasR: R Interface to the Keras Deep Learning Library, Journal of Open Source Software, 2(14), (2017) 296, doi:10.21105/joss.002961

[16] A. M. Taqi, A. Awad, F. Al-Azzo and M. Milanova, "The Impact of Multi-Optimizers and Data Augmentation on TensorFlow Convolutional Neural Network Performance," 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2018, pp. 140-145. doi: 10.1109/MIPR.2018.00032.

[17] Y. Zhu and S. Newsam, DenseNet for dense flow, IEEE International Conference on Image Processing (ICIP), 2017, pp. 790-794. doi: 10.1109/ICIP.2017.8296389.

[18] Riaz Ullah Khan, Xiaosong Zhang, Rajesh Kumar, and Emelia Opoku Aboagye, Evaluating the Performance of ResNet Model Based on Image Recognition, in: Proceedings of the 2018 International Conference on Computing and Artificial Intelligence (ICCAI 2018), Association for Computing Machinery, New York, NY, USA, 2018, pp. 86–90. doi:/10.1145/3194452.3194461.
[19] H. Kim, S. Park and J. Paik, Pre-Activated 3D CNN and Feature Pyramid Network for Traffic Accident Detection, 2020 IEEE International Conference on Consumer Electronics (ICCE), 2020, pp. 1-3, doi: 10.1109/ICCE46568.2020.9043125.
[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computerassisted intervention, Springer, 2015, pp. 234–241.
[21] D. Demirović, E. Skejić and A. Šerifović–Trbalić, Performance of Some Image Processing Algorithms in Tensorflow, 25th International Conference on Systems, Signals and Image Processing (IWSSIP), Maribor, Slovenia, 2018, pp. 1-4, doi: 10.1109/IWSSIP.2018.8439714.
[22] Ahmed Fawzy Gad, Practical Computer Vision Applications Using Deep Learning with CNNs With Detailed Examples in Python Using TensorFlow and Kivy, Apress Media LLC: Welmoed Spahr, 2018. 421 p.