# Classification of Chest X-Ray Images Using Deep Convolutional Neural Networks

Lara Laban, Radiša Jovanović, Mitra Vesović and Vladimir Zarić

***Abstract*—In this paper a method is proposed for the classification of a chest X-ray (*normal, pneumonia-consisting of bacterial and viral*) image data set. A deep convolutional neural network with an architecture resembling the VGGNet is presented using dropout, decay and data scaling. Since the dataset had a class imbalance, this was solved using a simple method called data scaling. The training of the neural network was done using small batches with a binary cross entropy loss function. The same neural network was then implemented adding batch normalization layers, and comparisons were made. Furthermore, the chest X-ray dataset was also trained using transfer learning with a pre-trained neural network VGG16 on the ImageNet dataset. Later on juxtapositions were made on using both techniques. Additionally, in applying these method we were able to achieve a satisfying classification for the training and validation datasets, save computational time and have a higher classification accuracy.**

***Index Terms*—convolutional neural networks; deep learning; transfer learning; batch normalization; chest X-ray dataset; image classification; dropout.**

## I. Introduction

Convolutional neural networks (CNNs) are a subset of deep neural networks, which are used for classifying images. The main idea is to take a set of images correctly labeled as the input data and used them to train our neural network so as to achieve an output with an appropriate categorization. The inspiration for CNNs comes from the observation of the animal visual cortex. Conversely, the flourishing of these networks only came recently due to the increase of computational power and the development of many possible libraries that could be used to battle complex mathematically based problems, such as back propagation. The first paper that introduced the convolutional neural networks as we have come to know them today has [1] demonstrated that a model which consists of a multilayered network can be successfully used for recognition of stimulus patterns according to the differences in their shapes. However, there is some debate that the true begging was when a paper in 1990 [2] demonstrated that a CNN model which aggregates simpler features into progressively more complicated features can be successfully used for handwritten character recognition. In 2012 the ImageNet Large Scale Visual Recognition Challenge [3], at that moment consisting of he 1000 categories and 1.2 million images received a submission that would propel the CNNs development once again. AlexNet [4] achieved a top-5 error of 15.3% , which at the moment surpassed by an astonishing 10% all of the other submissions, and had a much faster training time as it was implemented on a GPU. The following year, the same challenge, now with a larger dataset was won by ZFNet [5]. It had the top-5 error of 14.8%, however even more so important is that it was able to reduce the first layer filter size from $11 \times 11$ to $7 \times 7$ and had a stride of 2, rather than 4 in the pooling layer.

VGG16 is a convolutional neural network model proposed in the paper [6]. This model achieved 92.7% top-5 test accuracy. The main contribution of this model was that it used $3 \times 3$ kernel sized filters, instead of the $7 \times 7$ . It was trained for weeks using GPUs, and had a huge computational cost. However, it introduced a new idea using the same kernels throughout the entire architecture, this aided in generalization for classification problems outside of what they were originally trained on. If for a second we go back to LeNet [7] that was the foundation for all of these previously mentioned CNNs we can observe the main sequence of three layers convolution, pooling and non-linearity still play the key part, and sometimes it is beneficial not to import to many layers when training a smaller dataset [8]. Finally, in recent years transfer learning [9], which addresses crossdomain learning problems by extracting useful information from data in a related domain and transferring them for being used in target tasks, has been demonstrating a significant impact.

Pneumonia is one of the main causes of death amongst children, it was stated that 19% of all deaths of kids aged 5 years and less is connected with a viral or bacterial pneumonia [10]. Today, pneumonia is the single leading cause of mortality in young children according to the World Health Organization (WHO). An even scarier report made by WHO says that 95% of new-onset childhood clinical pneumonia occurs in developing countries, many of them located in Africa and South Asia [11].

The lungs of humans are made up of small sacs called alveoli, which fill with air when a healthy person breathes, in turn when a person with pneumonia breathes the alveoli which are filled in this case with pus and fluid, blocking the

Lara Laban is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: llaban@mas.bg.ac.rs)

Radiša Jovanović is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: rjovanovic@mas.bg.ac.rs)

Mitra Vesović is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: mvesovic@mas.bg.ac.rs)

Vladimir Zarić is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: vzaric@mas.bg.ac.rs)

oxygen from arriving to the lungs, limit the capacity to intake oxygen. One of the main ways to have a proper diagnosis is radiographic data. X-rays can help in distinguishing between different types of pneumonia. However, since rapid interpretation of images is sometimes very hard, especially in developing countries, new methods are always sought after and proposed.

For that reason, an idea to battle this kind of a problem was proposed in a brilliant paper [12] published in 2018 which suggested image-based deep learning to identify various medical diagnoses, including the chest X-ray images from children. Using convolutional neural networks, transfer learning to be precise, they achieved an accuracy of 92.8% in distinguishing between normal and pneumonia images, over the course of 100 epochs.

This paper is organised in the following manner: section 2 represents a description of a dataset which is used in the training and validation of the proposed neural network. In section 3 the main methods which are used are explained in detail, as well as the architecture of the CNN. As a result, in section 4 we discuss the results and compare the methods, based on accuracy and loss functions. In section 5, following a short summary a conclusion is made and future work and possible directions are stated.

## II. Dataset and its implementation

The dataset which is used in this paper consists of 5856 chest X-Ray images from children [13], including 4392 pneumonia ray (*bacterial and viral*) and 1464 normal. Being than the dataset consist of a couple of thousand pictures, there is no need to take an approach of data augmentation, where we increase the diversity of data by altering the original samples using translation, rotation, shearing, flips and adding them to the training set. However, we observe that the pneumonia part of the dataset is much larger than the part of the normal set, almost 4 times as big, resulting in a class imbalance. One way to correct this, so that our neural network may learn appropriately and not pick the pneumonia label naturally is to scale the data. This can be done by computing a weight for each class during the training, resulting in an array $[1,3]$, and as an outcome amplifying the loss by a larger weight when we approach normal data. In this example treating an instance of normal as 3 instance of pneumonia, aids in this disproportion.

During the preprocessing of images we resized all the images to a fixed size $64 \times 64$, and in doing so we also maintained the aspect ration. The reasoning behind this being that all the images in a dataset need to have a fixed feature vector size. This means all the images will have identical widths and heights, making it easer to quickly load and preprocess a dataset and briskly move through our convolutional neural network. The aspect ratio will enable us to resize the images along the shorter dimension, be it width or height, and in cropping it, will maintain the ratio. It is important to note that this step is not necessary if you are not working with a difficult dataset. Notwithstanding its benefits, it was implemented in this particular dataset.

### A. ImageNet dataset

ImageNet is a dataset consisting of over 14 million images, which belong to one thousand classes. It was used as the dataset in the highly respected convolutional neural network model VGG16 which was proposed by Oxford scientists. In this paper the VGG16 network was used as a pre-trained convolutional neural network, in order to incorporate transfer learning and compare it to the original paper, mentioned beforehand, as well as the architecture that we propose.

## III. Methods description

In order to try and reduce overfitting and increase our classification accuracy on the chest X-ray dataset we endeavor in performing two types of neural network training techniques:

- dropout and decay (with and without batch normalization),
- transfer learning (neural networks as feature extractors)

The first technique that is used in order to improve the generalization error in the convolutional neural network is dropout [14]. Dropout is nothing more than a form of regularization, which succors us in controlling the model capacity. Furthermore, it increases validation accuracy, sometimes at the expense of the training accuracy in order to help avert overfitting. Simply put, some number of layers outputs are dropped, making the current layer have a different number of nodes, as opposed to the previous layer. This helps our model to observe multiple nodes and activate them when similar inputs are provided, therefore, directly aiding the generalization problem. The dropout layers are arranged in the network in such a manner that we have randomly disconnected nodes by a probability of 0.25 in the first few layers; and with a double increase in probability in the last layer. The reason for this is that if the first layers are dropped by a higher probability, then that will later affect the training. The dropout is implemented after the pooling layer, and before the next convolutional layer (or last flatten and dense layers). Decay that is used in this neural network is a standard decay that can be obtained using the Keras library in Python. Since the learning rate $\alpha$ controls the step that is made along the gradient, larger steps are usually used in the beginning to make sure that we do not stagnate in the local optima, while smaller steps are used deeper in the network and near the end of the convolution in order to converge to a global minimum. We have initialized the learning rate to be 0.05, and applied the following formula to adjust it after each epoch,

$$\alpha_{i+1} = \frac{\alpha_i}{1 + k \cdot i} \tag{1}$$

where $\alpha$ is the current learning rate, $i$ is the epoch and $k$ is the decay calculated as the division between the learning rate and the number of epochs. This type of adjustment of the learning rate each epoch, can increase accuracy, as well as reduce the loss function and the time necessary to train a network. Batch normalization [15] is used to normalize the

activations of a given layer's inputs by applying mean and standard deviation before passing it onto the next layer. In addition, the covariate shift refers to a change in the distribution of the input variables which are present in the training and validation data. Since it has been proven that the training of the neural network is the most coherent when the inputs to each layer are alike, the main intention is that even when the explicit values of inputs layers to hidden layers change, their mean and standard deviation will still remain relatively the same, thus reducing the covariate shift. Batch normalization has demonstrated an immensely effective approach to reducing the number of epochs necessary for training by allowing each layer to learn independently. Here the idea that differs from the original paper and is first proposed in [16] states that the batch normalization should be implemented after the activation layer. The main reasoning behind this is that we want to avoid setting the negative values coming out of the convolution layer to zero. Instead we pass them trough the batch normalization layer, right after the activation (ReLU) layer, and assure that some of the features that otherwise would not have made it do. This yields a higher accuracy and lower loss, and is to this day a debate amongst the creators of Keras.

Finally, the second technique is [17] transfer learning, a machine learning technique where networks can behave as feature extractors. Transfer learning is nothing more that the ability to use a pre-trained model to learn patterns from data, on which the original network was not trained on. As previously stated deep neural networks trained on a large scale dataset ImageNet have demonstrated to be superb at this task.

When treating networks as feature extractors we choose a point, in this case before the fully connected layer and remove it. Subsequently, in this particular example while using the VGGNet pre-trained on the ImageNet we removed the fully connected layer and stopped at the last pooling layer where the output shape is $7 \times 7 \times 512$, 512 filters with the size $7 \times 7$. Now, our feature vector has $7 \times 7 \times 512 = 25088$ values and it will be used to quantified the contents of the images, which were not included in the original training process. The format which allows us to extract these features is the hierarchical data format version 5 (hdf5), which is used to store and organize large amount of data.

Transfer learning is an optimization, which has been proven to yield a better performance and drastically save time. This is precisely why we used it in this paper, to see if we could obtain a higher classification, and perform faster. Transfer learning relaxes the hypothesis that the training data must be independent and identically distributed with the test data, which we clearly stated as a must in the beginning of this chapter. Moreover, transfer learning is able to solve the problem of insufficient training data. Furthermore, there is the option to remove the fully connected layers of the existing network in order to add a new fully connected layer to the CNN and fine tune the weights to recognize object classes. However, here it was not implemented since treating networks as arbitrary feature extractors was enough.

In the following sections we will demonstrate the architecture of a CNN that is based on VGGNet, its implementation with and without batch normalization, and additionally transfer learning will be presented instead of the CNN that was previously explained.

### *A. Convolutional Neural Network architecture*

Into the bargain all that was explained, we picked the following CNN architecture shown in Fig. 1. It is consisted of multiple convolutional and pooling layers, as well as the fully connected layers. The first two convolutional layers learn 32 filter each with a size $3 \times 3$.

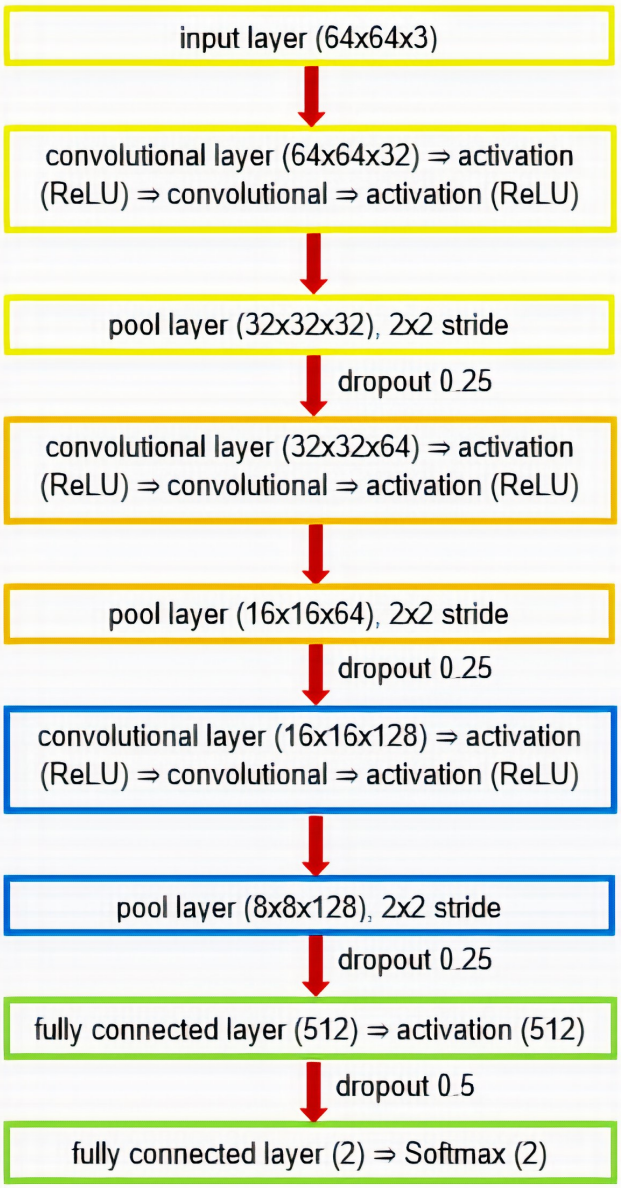


Fig. 1. A schematic of the convolutional neural network without batch normalization, that resembles the VGGNet. All of the convolutional layers that precede the fully connected layers have filters 32,64,128 that are the same size $3 \times 3$. The probability distribution is applied in the last layer using Softmax and the output yields two class labels normal and pneumonia.

Sequentially, the fourth and the fifth layers learn 64 filters with the size $3 \times 3$ and the last two learn 128 filters with the size $3 \times 3$. The pool layer is used to reduce the computational load and the number of parameters, thus reducing the risk of overfitting. We used a max pooling layer with a pool size $2 \times 2$ and a stride 2. Finally, we have the fully connected layer which consists of 8192 parameters, input values which learn 512 nodes. The activation layers which were used are Rectified Linear Unit (ReLU) defined as,

$$f(x) = \max(0, x) \tag{2}$$

where x is the input into the neuron.

Softmax or the normalized exponential function assigns normalized class probabilities for each prediction, and is represented by,

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}} \tag{3}$$

for $i = 1, ..., k$ and $\mathbf{z} = (z_1, ..., z_k) \in \mathbb{R}^k$.

Softmax takes an input vector and normalizes it into a probability distribution between $[0,1]$. Therefore the sum of all output values is equal to 1, which in turn makes the training converge more quickly. In order to achieve this, before training we must include one hot encoding in order to convert the labels from integers to vectors.

In addition, later when we want to add the batch normalization layer, we can apply it after each activation layer, as discussed previously.

### *B. Implementation and training of a much simpler version of the VGGNet*

Taking into the bargain all that was explained before, the implementation of this CNN was done by using the Python programming language.

We used Keras [18] which is mainly used for implementing of activation functions, optimizers, convolutional and pooling layers, and is actually able to do backpropagation automatically. Further, we used TensorFlow [19] an open-source library developed by the Google Brain team, with the main purpose to help combat convoluted numerical computations and large-scale machine and deep learning. The main advantage of TensorFlow is its compatibility with Sckit-Learn and Keras library.

Right after we load and preprocess our images dataset it is necessary to use one hot encoding. This is done by using a part of the Sklearn library LabelBinarizer. However beforehand we must split the training data and the validation data, here we opted to split it 75% and 25%, sequentially. The next step is the implementation of an optimizer, here we used the stochastic gradient descent (SGD) optimizer. The SGD optimizer was set to a learning rate of $\alpha = 0.05$, with a decay in order to slowly reduce the learning rate over time and converge to the global solution more efficiently. Decaying the learning rate is beneficial in reducing overfitting and obtaining a higher classification accuracy. The smaller the learning rates are, the smaller the weight update will be enabling us to converge. The gradient descent method is an iterative optimization algorithm that operates over an optimization surface. It is a simple modification to the standard algorithm of gradient descent. The main purpose of SGD is to calculate the gradient and adjust the weights of the training data (but not on the whole dataset, but rather on a mini batch). The mini batch method is a blend of the SGD and batch methods were the neural network selects a part of the training data and updates the weights, but trains the network with the average weight update. Usually the smallest standard batch size which is used is 32, however we opted to use 24, as it complemented our data. The reasoning behind this is that present research confirms that using small batch sizes achieves the best training stability and generalization performance, for a given computational cost, across a wide range of experiments. The loss function which was used is the binary_crossentropy function. This was done because we only had two classes, if there were more we would have had to use categorical_cross_entropy, but have in mind we could have used categorical as well, but studies show that binary is much more efficient in this case.

The training was done on 30 epochs since it was enough to achieve satisfying results. After the training we implemented a method that takes the weights and the state of the optimizer and serializes them to the disc in a hdf5 format, in order to load them and test the labeling.

### *C. Implementation using transfer learning*

The first step in this process is to extract features from VGG16, in doing so we are forward propagating the images until a given layer, and then taking those activations and treating them as feature vectors. Here the main two differences are that we used the standard a batch size of 32 and the training and test split is done at the same time as training, we again split it into 75% training data and 25% test data. Once the extraction of the features was done, we trained the classifier on those features. We also implement the GridSearchCV class to assists us to turn the parameters to the LogisticRegression classifier.

The final results are presented in the following chapter, comparisons are made and a visual representation of the graphs is shown using Matplotlib in order to estimate if there is overfitting.

## IV. RESULTS AND COMPARISONS

The results of the CNN without batch normalization are presented in Table 1. We clearly see that our neural network has classification accuracy of 95%. In the following tables we use the term precision which represents true positive divided by a sum of true positive and false positive; recall which represents true positive divided by a sum of true positive and false negative. Therefore, precision is good to determine when the cost of false positives is high, on the other hand recall tells us the number of correctly labeled data. Ultimately, we have the f1-score used to find the weighted average of recall and precision.

TABLE I
CNN WITHOUT BATCH NORMALIZATION: RESULTS

| | precision | recall | f1-score |
|---|---|---|---|
| normal | 0.93 | 0.91 | 0.92 |
| pneumonia | 0.97 | 0.97 | 0.97 |
| | | | |
| accuracy | | | 0.96 |
| macro avg | 0.95 | 0.94 | 0.95 |
| weighted avg | 0.96 | 0.96 | 0.96 |

In analyzing the curves shown in Fig. 2 we see that our network learned until the 30 epoch, beyond that was simply not necessary since we already had excellent results. We can also observe that our loss and accuracy curves both almost match for training and validation, with slight deviations.

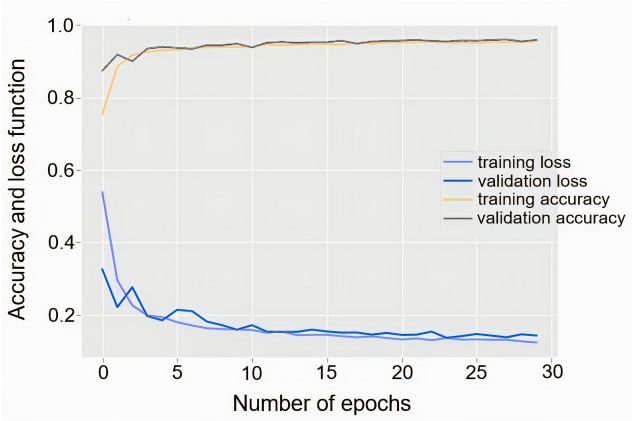


Fig. 2. A graph depicting a convolutional neural network without batch normalization, that resembles the VGGNet – training and validation loss and accuracy curves

In Fig. 3 we can see how the labeling looks, when we use the trained and saved model to label the data with this obtained accuracy.

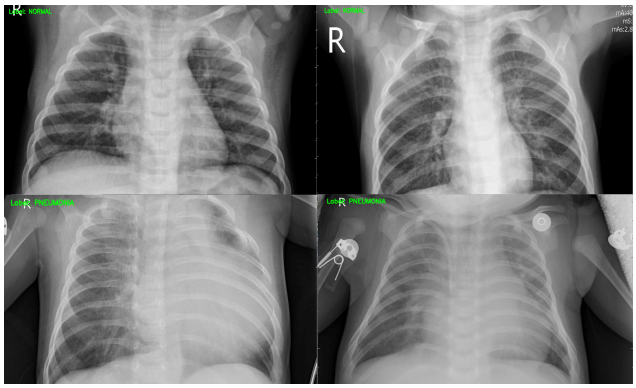

Fig. 3. The pre-trained CNN weights are loaded from the disk and make predictions for 30 randomly selected images. In the upper left and right corner we have an example of normal lungs, and in the lower left and right corner an example of pneumonia lungs.

TABLE II
CNN WITH BATCH NORMALIZATION: RESULTS

| | precision | recall | f1-score |
|---|---|---|---|
| normal | 0.93 | 0.92 | 0.92 |
| pneumonia | 0.97 | 0.97 | 0.97 |
| | | | |
| accuracy | | | 0.96 |
| macro avg | 0.95 | 0.95 | 0.95 |
| weighted avg | 0.96 | 0.96 | 0.96 |

In Table 2 we see that the CNN with batch normalization obtained the same classification accuracy of 95% after 30 epochs. However, in analyzing the curves shown in Fig. 4 we see that our network learned until the 30 epoch, because further training past epoch 30 would result in overfitting and a wider generalization gap (loss function - the gap between the training loss and validation loss).

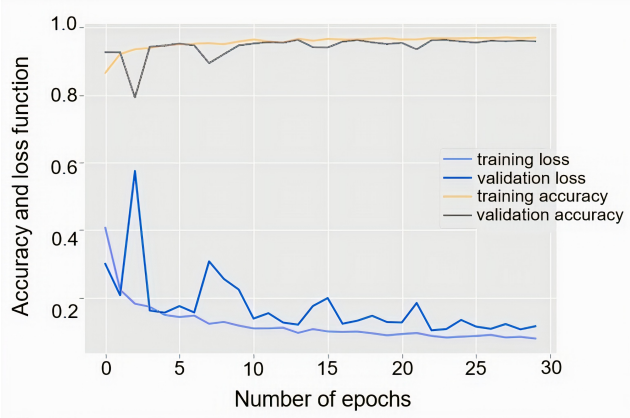


Fig. 4. A graph depicting a convolutional neural network with batch normalization, that resembles the VGGNet – training and validation loss and accuracy curves

TABLE III
TRANSFER LEARNING USING VGG16

| | precision | recall | f1-score |
|---|---|---|---|
| normal | 0.96 | 0.93 | 0.95 |
| pneumonia | 0.97 | 0.98 | 0.98 |
| | | | |
| accuracy | | | 0.97 |
| macro avg | 0.97 | 0.96 | 0.96 |
| weighted avg | 0.97 | 0.97 | 0.97 |

In Table 3 we can see the results obtained by using transfer learning have a classification accuracy of 97%, which is by far the best. Furthermore, we observe that the CNN with batch normalization had a higher recall and a problem with overfitting past epoch 30, therefore the CNN without it seams like a better choice. Nevertheless, it is clear then when taking into account all three approaches we shall choose transfer

learning, because not only does it yield a higher classification accuracy, but it also wasted less computational time.

## V. Conclusion

In this paper we described two different approaches of using convolutional neural networks too classify a dataset consisting of normal and pneumonia infected lungs. We used a CNN that we constructed based on the VGGNet and implemented it with and without batch normalization. Furthermore, we used a transfer learning technique by extracting features of the neural network VGG16 trained on the ImageNet dataset. The main idea of this paper was to see if a different approach can have better results on this particular dataset, as well as see if a smaller neural network could have almost as good classification, as transfer learning. The final results, when compered had a higher classification accuracy by a couple of percentages, and also achieved so in just 30 epochs, as opposed to 100 epochs, so we can conclude the goal was obtained.

Further research will focus on implementing different types of optimizers, including metaheuristic algorithms as optimizers. Also, we will focus on battling larger datasets and obtaining high classification accuracy using various methods.

## Acknowledgment

This paper was conceived within the research on the project: "*Integrated research in the field of macro, micro and nano mechanical engineering - Deep machine learning of intelligent technological systems in production engineering*", The Ministry of Education, Science and Technological Development of the Republic of Serbia (contract no. 451-03 - 68 / 2020-14 / 200105), 2020.

This work was financially supported by the Ministry of Education, Science and Technological Development of the Serbian Government, Grant TR-35029 (2018-2020).

## References

[1] K. Fukushima, S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455-469, 1982.

[2] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, L. D. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in Neural Information Processing Systems 2*, pp. 396-404, June, 1990.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Haung, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, April, 2015.

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, pp. 1097-1105, 2012.

[5] M. D. Zeiler, R. Fergus, "Visualizing and understanding convolutional networks," 13th European Conference, Zurich, Switzerland, pp. 818-833, September 6-12, 2014.

[6] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[7] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, n. 11, pp. 2278-2324, November, 1998.

[8] Z. Li, W. Yang, S. Peng, F. Liu, "A survey of convolutional neural networks: Analysis, applications and prospects," arXiv preprint arXiv:2004.02806, 2020.

[9] L. Shao, F. Zhu, X. Li, "Transfer learning for visual Categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019-1034, May, 2015.

[10] I. Rudan, C. Boschi-Pinto, Z. Biloglav, K. Mulholland, H. Campbell, "Epidemiology and etiology of childhood pneumonia.", *Bulletin of the World Health Organization*, vol. 86, no. 5, pp. 408–416, May, 2008.

[11] World Health Organization, Pneumonia, https://www.who.int/news-room/fact-sheets/detail/pneumonia, (last accessed 06/07/2020).

[12] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, M. Y. L. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. N. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, K. Zhang, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122-1131, February, 2018.

[13] Public datasets; Chest X-Ray Images (Pneumonia), Version 2 https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia, (last accessed 09/03/2020).

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", arXiv preprint arXiv:1207.0580, Jul, 2012.

[15] S. Ioffe, C.Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", Proceedings of the 32nd International Conference on Machine Learning, vol. 37, pp. 448-456, July, 2015.

[16] A. Rosebrock, *Deep Learning for computer vision with Pyhton: Starter Bundle,* 1st ed. PyImageSearch, 2017.

[17] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, "A survey on deep transfer learning," 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018.

[18] Keras; Python Deep Learning Library https://keras.io, (last accessed 09/03/2020).

[19] TensorFlow; An end-to-end open source machine learning platform for everyone. https://www.tensorflow.org, (last accessed 09/03/2020).