ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20191023102807

Original scientific paper

IMPLEMENTATION OF INTELLIGENT MODEL FOR PNEUMONIA DETECTION

Željko KNOK, Klaudio PAP, Marko HRNČIĆ

Abstract: The advancement of technology in the field of artificial intelligence and neural networks allows us to improve speed and efficiency in the diagnosis of various types of problems. In the last few years, the rise in the field of convolutional neural networks has been particularly noticeable, showing promising results in problems related to image processing and computer vision. Given that humans have limited ability to detect patterns in individual images, accurate diagnosis can be a problem for even medical professionals. In order to minimize the number of errors and unintended consequences, computer programs based on neural networks and deep learning principles are increasingly used as assistant tools in medicine. The aim of this study was to develop a model of an intelligent system that receives x-ray image of the lungs as an input parameter and, based on the processed image, returns the possibility of pneumonia as an output. The implementation of this functionality was implemented through transfer learning methodology based on already defined convolution neural network architectures.

Keywords: computer vision; machine learning; neural networks; pneumonia

1 INTRODUCTION

Nowadays, machine learning and artificial intelligence methods are culminating in every aspect. These methods are increasingly being used in business systems, enterprises of various types and in science, in order to improve efficiency of the facilities they serve. While numerous works focus on a conscious form of artificial intelligence that would somehow replace humans, computers with learning ability have been around for some time.

Object recognition and deep analysis are just some of the characteristics of machine learning and neural networks. Although the concept of neural networks was developed back in the 20th century, training process represented hard job for computers from that time. With the development of hardware, GPU (Graphical Processing Unit) and CPU (Central Processing Unit), and high availability of data, neural networks are experiencing significant upswing. Sometimes the diagnosis of the disease was almost exclusively dependent on the experience of the doctor and his assessment. Today, more and more diseases can be diagnosed with the help of some kind of computer or machine.

This paper is based on the diagnosis of pneumonia according to the obtained x-ray image as an input parameter. With the advancement of artificial intelligence, it can be very useful in the diagnosis of x-ray images and play an important role in the detection of disease, serve as a computer assistant to radiologists, and increase their efficiency in diagnosis.

Particularly noteworthy is the application of AI system for the diagnosis of pediatric pneumonia using chest X-ray images. This tool may eventually aid in expediting the diagnosis and referral of these treatable conditions, thus facilitating earlier treatment, resulting in improved clinical outcomes. [1]

The aim of this project is to implement software for the automatic detection of pneumonia and the use of computer algorithms in the field of machine learning to automate the process of obtaining the most accurate diagnosis, which could reduce the possibility of errors and misdiagnosis that can lead to unwanted consequences. Project uses several different technologies combined. The implementation of the convolution neural network model was done in Python using Anaconda as programming environment. [2]

2 MACHINE LEARNING

Machine learning is a branch of artificial intelligence that focuses on designing algorithms that improve their own performance based on input. Machine learning is one of the most active and represented areas of computing and is the base of data science.

It enables computers to learn in a manner similar to humans, that is, to gather knowledge based on past experience and analysis. Instead of constantly updating the program code, the computer eventually becomes independently capable of improving the performance of its algorithms. Data processing using machine learning methods often results in a model capable of performing some kind of prediction on later test data. [3]

2.1 Deep Learning

In this paper, the focus is on deep learning, which can be characterized as a branch of machine learning inspired by the structure of the human brain. This principle was developed by modeling the first neural networks back in the 1940s, with the term first mentioned by Rina Dechter in 1986 and defined as a field of machine learning based on the presentation of data to complex representations at a high level of abstraction to which comes by learned nonlinear transformations.

Deep learning methods are most commonly used in challenging areas where the dimensionality and complexity of the data is extremely high. Deep learning models mainly carry out learning process on a large set of data, which falls within the scope of supervised learning, and form the backbone of today's principle of autonomous driving, disease detection and pattern recognition, with result that were impossible before. [4]

2.2 Biological Neural Networks

The human brain consists of densely interconnected nerve cells that serve to process data, also called neurons. Within the human brain, there are approximately 10 billion neurons and approximately 60 billion connections between them.

A neuron as a unit represents a very simple structure, but a large number of neurons represent tremendous processing power. One neuron consists of a soma, representing the body of a cell, fibers called dendrites, and one longer dendrite, or axon. Dendrites are located in a network around the cell of that same neuron, while the axon extends to the dendrites and cells of other neurons. Signals or information are transmitted from one neuron to another by complex electro-chemical reactions. Chemicals are released from synapses and cause a change in electrical potential in neuronal cells. As the electric potential threshold is reached, the electrical impulse sends the action potential over the axon. The impulse extends until it reaches a synapse and increases or decreases its potential.

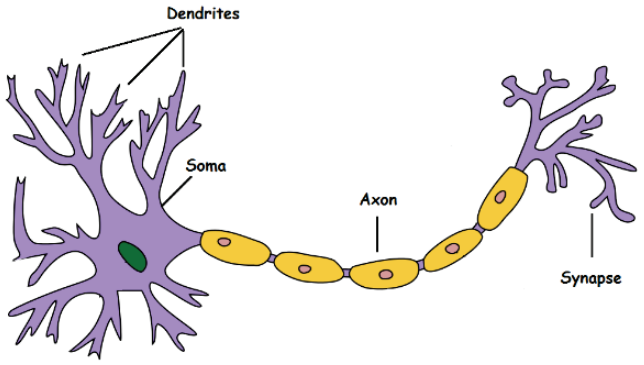

Figure 1 Structure of a biological neuron

2.3 Artificial Neural Networks

The artificial neural network, inspired by the work of the human brain, contains a number of connected processors, neurons, which have the same role as biological neurons within the brain. They are connected by links whose signals can pass from one neuron to another, thus transmitting important information. Each neuron receives a certain number of input signals in the Xi tag. Each connection has its own numerical value, namely the weight of the Wi, which is the basis for long-term memory in artificial neural networks. The Xi and Wi values are summed by the transfer or activation function, and the result is sent as output Y to another neuron.

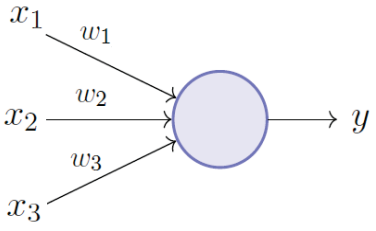

Figure 2 Structure of artificial neuron

In artificial neural networks, neurons are arranged in layers that are interconnected by the links through which the signals pass, that is, information that can be of great importance for the end result and performance of the neural network. Connections between neurons are triggered if the condition is met. Condition is defined with activation function that will be elaborated later.

The layer that receives information from the environment is called the input layer. It is associated with the hidden layers in which information is processed, that is, the network learns. The last layer that produces outputs is called the output layer. The learning process is performed by changing the value of the weights or connections between the neurons, comparing the required and obtained sizes on the output layer and calculating the error.

Based on the error value obtained, it is attempted to be reduced by each subsequent step, and this is done by weight correction based on a defined learning rule related to data acquisition.

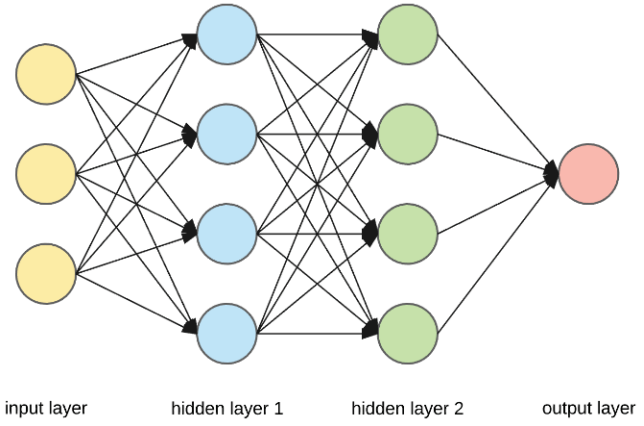

Figure 3 Layers of artificial neural network

2.4 General Learning Rule

Each artificial neural network is based on a general learning rule that involves collecting relevant data, which is then divided into training and validation data.

After the data collection is completed, it is necessary to determine the architecture of the neural network, which involves determining the number of layers in the network and the number of neurons in each layer, and then selecting the type of connection between the neurons together with the learning rule that is the basis for defining criteria that determines the architecture of the neural network. The next step is learning, which is the basis of artificial neural networks. Learning involves initializing weights, training a model based on a training dataset, and checking the amount of error that weights are corrected after each iteration, which refers to going through all the training samples, called the epoch.

Learning lasts until the desired number of epochs is reached or a wanted error is met. In the initial stages of learning, the neural network will adapt to the general trends present in the data set, so the error in both the training set and

the validation set will fall through time.

During the longer learning process, there is a possibility that the neural network can begin to adapt to the specific data and noise of the learning data, thereby losing its generalization property. The error on the training set will drop while on the validation set it will start to grow. The moment the validation set error starts to grow, the learning process must be interrupted so that the model does not become overfitted.

With the completion of the learning process, it is necessary to test the operation of the network using previously obtained validation data. The difference between the learning and the testing phase is that the neural network no longer learns in the testing phase and the weight values are fixed. The evaluation of the network is obtained by calculating the error and comparing it with the desired error value. If the error is greater than allowed, additional training data may need to be collected or the number of epochs increased for better results, since in this case the network is unsuitable for use.

2.5 How Neural Networks Work

In working with neural networks, there are two basic algorithms used today. These are the feedforward algorithm and the so-called backpropagation algorithm.

Working with neural networks allows us to define complex, nonlinear hypotheses consisting of one or more neurons that can be arranged in one or more layers that build the neural network architecture. This method of mapping the input vector to the network outputs is called forward propagation.

Backpropagation algorithm is one of the main reasons that made artificial neural networks known and usable for solving different types of problems. With this algorithm, artificial neural networks have been given a new capability, supporting multiple layers, and the backpropagation algorithm has proven to be the most common and effective method in deep neural networks. The algorithm is used in conjunction with optimization methods such as gradient descent. The principle of network operation is based on the transfer of input values from the input to the output layer, determination of the error and propagation of the error back through the neural network from the output to the input in order to train the network as best as possible and to reduce the existing error and thus improve the final results.

2.6 Types of Activation Functions

There are many types of activation functions that determine whether and how neurons are activated within a network. Activation functions can be divided into linear and nonlinear, and in practice, only a few that have proven useful are used. Linear functions are used in regression problems when unlimited output of any kind is required in the output layers, while nonlinear activation functions are more suitable for working with classification problems where the outputs are limited to small quantities.

The most popular functions to use within classification problems are jump functions and sigmoidal functions.

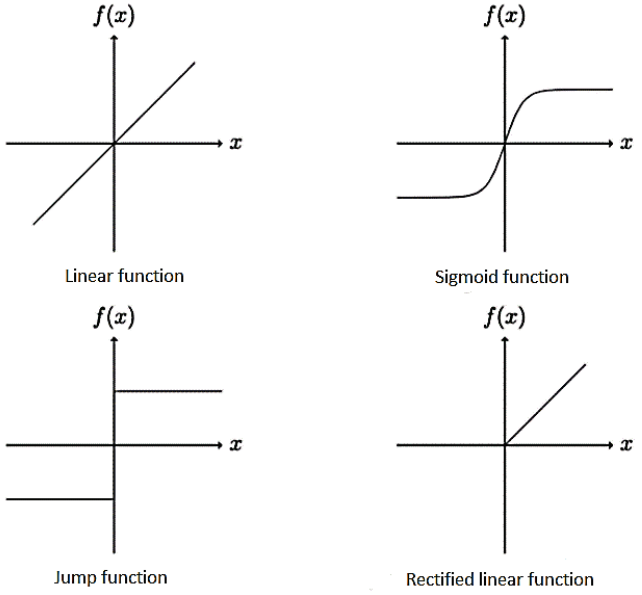

Figure 4 Types of activation functions

2.7 Convolutional Neural Networks

In the neural network structures mentioned, each neuron output was a scalar value. Convolutional neural networks are a special type of artificial neural networks for processing unstructured data such as images, text, sound and speech in which extensions occur in the form of convolution layers. The outputs from the convolution layers are two-dimensional and are called feature maps. The input to convolutional neural networks is two-dimensional (image data), while kernels are used instead of weight values. In addition to the convolution layers mentioned, these types of neural networks have specific pooling layers and fully connected layers.

Convolutional neural networks usually start with one or more convolution layers, followed by a pooling layer, then the convolution layer again, and the process is repeated several times.

The convolution layer is a fundamental part of any convolution neural network. Each convolution layer consists of filters containing weight values that need to be learned in order for the network to return better results.

In the initial phase, the input convolves (multiplies) with the filter and produces a scalar product over the entire width and length of the image. The result of the convolution is a two-dimensional activation map that shows the filter response at each position in the array. In this process, the network will learn the weights within the filter to activate the filter where it recognizes certain image properties, edges, shapes, and the like. In order to define the exact size of the output of the convolutional layer, it is also necessary to define the stride of the filter according to the input data. The stride determines how far the filter will move in height and width during the convolution process.

The output width is indicated by W_y and the input width by W_x. The filter stride is denoted by the letter S, while F

indicates the square filter size. The output width after the convolution can be defined as:

$$W_y = \frac{W_x - F}{S} + 1, \tag{1}$$

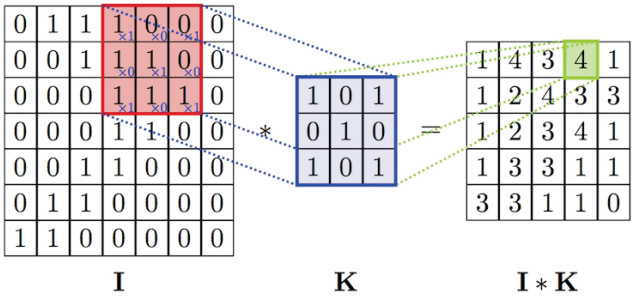

Figure 5 Convolution process

The pooling layer contains a filter that reduces the dimensions of an image. In a convolutional neural network, pooling layer is most commonly used after several convolution layers in order to reduce the resolution of maps generated by the convolution layer. The pooling layer filter is different from the convolution layer filter because it does not contain weight values. The specified filter is used to select values in the filter default dimensions.

Of the several types of pooling, the most commonly used are average pooling and max pooling. Average pooling replaces the arithmetic mean of clustered values, while the max pooling simply selects the maximum value. The benefit of max pooling is to store the stronger and more prominent pixels in the image, which are more important for getting the end result, while the irrelevant pixels are eliminated.

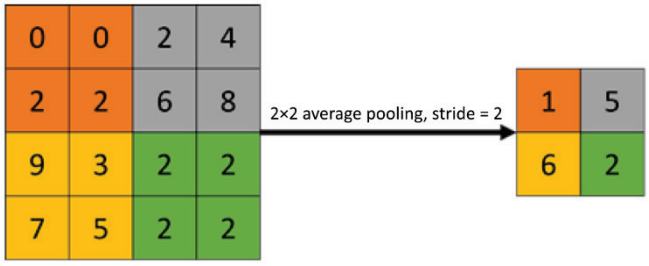

Figure 6 Average pooling

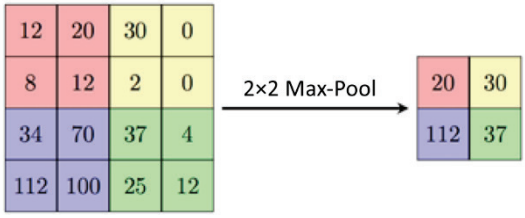

Figure 7 Max pooling

In the pooling layer, the dimensions of the output maps are calculated as:

$$D_n = \frac{D_s - F}{S} + 1, \tag{2}$$

where D_s is the old dimension, F is the filter width, and S is a jump between 2 value selections.

The fully connected layer is in most cases used in the final layers of the network. The reason for using fully connected layers is to reduce the dimensions of the image by passing it through the neural network, since complete connectivity is defined as the square number of connections between layers.

For example, for image data with dimensions 200×200×3, the input layer would have 120,000 input values. If we fully linked this to a hidden layer consisting of 1000 neurons, we would have 120 million weight values to learn, which requires big computing power. This is why fully connected layers are used in the later stages of the neural network.

2.8 Avoiding Local Minimums

During the learning process of the neural network, the goal is to find the location of the global minimum of error, which means that the model is at the best possible level at a given moment, and the learning process can stop. In this process, the so-called local minimums can fool the network in a way that the network thinks it is within the global minimum. Avoiding local minimums can be achieved by using various methods.

Known methods for avoiding local minimums:

Random Transformations – Random transformations serve to augment an existing training dataset that is accomplished by operations such as translation, rotation, and scaling. This increases the number of data without the need to collect additional samples. An increased amount of learning data makes the network less likely to get stuck in local minimum. Random transformations can be performed during each iteration in the learning process or by pre-processing data before training begins.

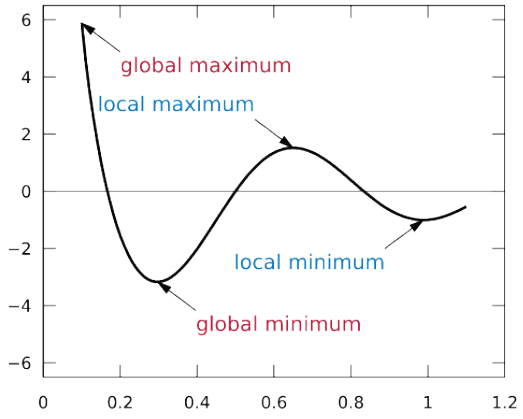

Figure 8 Local and global minimum comparison

Dropout – During every iteration of the learning process, some neurons are "accidentally" switched off, thus creating the appearance of a large number of architectures, although in reality they only work with one. In the case of multiple architectures, it is unlikely that they will be stuck at the same local minimum.

2.9 Transfer Learning

Conventional deep learning algorithms are traditionally based on isolated tasks, while a single neural network serves a particular type of classification. Transfer learning is a new methodology that seeks to change this and circumvent the paradigm of isolated learning by developing knowledge transfer methods that can use models learned on one type of classification for multiple different tasks.

In this way, a model initially created for one type of problem can later be used as a starting point for solving a new type of classification, and thus give better results than initializing a new neural network from the beginning.

An analogy can be made with learning to drive in real life. In this analogy, learning how to drive a motorcycle can be greatly assisted by the knowledge of driving a car. Transfer learning works in a similar way.

Transfer learning is nowadays a very popular approach on which the practical part of this paper is based. Choosing pre-trained models and already defined neural network architectures can be of great use in solving complex problems such as detecting pneumonia on x-ray images.

After the architecture of the existing neural network with defined layers has been loaded, it is necessary to delete the last, output layer from the specified network, and replace it with a new output layer related to the problem for which we will use the network.

The next step is to conduct training on the complete, already defined, architecture and all layers of the network on the learning dataset. By using this way of learning, the neural network will be able to apply the classification principles learned in the previous tasks to a new type of problem, and in that way the results will be better without the need to define layers and create a new neural network from the beginning.

Transfer learning is good to use in cases where we do not have a large dataset to learn. In the case where we have approximately 1000 images to perform learning process, by merging that 1000 data with trained networks that have been trained with millions of data, we can gather many learned principles for sorting and classification, and in that way improve model efficiency and reduce training time.

VGG16 is a convolution neural network architecture proposed by K. Simonyan and A. Zisserman of Oxford University. The model achieves 92.7% accuracy on the ImageNet image dataset, which consists of over 14 million image data divided into 1000 classes.

The VGG16 network architecture consists of 13 convolution layers in which the number of filters increases from 64 to 512; 5 compression layers with highest value compression; and, 3 fully connected layers that are used to avoid local minimums using the dropout technique described earlier.

Big progress in comparison with previous neural network architectures has been made after resizing the convolution filter to 3×3.

The filter of the pooling layers is 2×2 in size with a stride of 2, while all the hidden layers of the network use the rectification linear activation function described in the previous chapters. All convolutional layers use stride and padding of value 1, and the total number of parameters of the specified network architecture is 138 million. [5]

3 MODEL IMPLEMENTATION

This chapter describes the implementation of the neural network model. The process of collecting image data for learning, division of the set into a learning and validation set, visual analysis and comparison of data for learning and with data for validation, comparison of the number of positive and negative images, and preprocessing of the collected data for entering the learning process are presented.

The procedure of retrieving the previously described architecture of the VGG16 network and changing the output layer in accordance with the collected data were developed, the learning curves using Tensorboard technology and the implementation of the training or learning process were presented. After learning was completed, the best-performing model was saved and an evaluation of that model was performed to graphically present prediction accuracy on validation data not used in the learning process.

3.1 Data Collection

The first step in implementing a model for detecting pneumonia on x-ray images is to collect imaging data that will consist of sets for training and validating the model. Dataset available for download on the Kaggle site was used. Kaggle offers users a large number of different datasets that can be used for various research purposes.

The dataset selected consists of a total of 5863 lung x-ray images divided into two categories. Existing categories are x-rays with a positive diagnosis of pneumonia and images of normal, healthy lungs with no indication of disease. The set contains 1583 images of healthy lungs and 4273 images positive for pneumonia. The image data format is .jpeg, while the image dimensions are different and vary from image to image. In the later steps, it is necessary to change the image dimensions that will be supported as input for the VGG16 network architecture. [6]

3.2 Data Preprocessing

After successful process of the data collection for training process, before learning, the same data should be processed in such a way that they are suitable for entering the network.

Image file paths were originally loaded using a function from the Scikit-learn library that loads the paths of all files in a given directory as parameters of those directories within the main directory as categories of individual files, or their paths.

Subsequently, categories or labels are defined from the obtained data set, depending on whether the data is in the subdirectory of images with pneumonia or images without disease. Saving image categories was done using functionality from the Numpy library.

By completing the label definition for all retrieved image paths, the total image dataset should be divided into a learning set and a set for validation. In this case, 90% of the

total data set was determined for learning purposes, with the remaining 10% for validation.

After splitting paths and labels into training and validation data, functions were implemented to load saved paths and convert them to image data and resize images to 256×256, which will be required for later input to the initial layer of the neural network.

3.3 Data Visualization

This chapter elaborates a section based on the visualization of preprocessed data for an easier idea of the dataset collected, the ratios between learning and validation data, and the like. Graphics and diagrams to show the ratio between the data were implemented using the Matplotlib library.

A visualization feature that has been implemented offers a graphical representation of the data. This function performs visual processing of the data ratio, namely, the ratio between the learning and the validation sets and the number of images positively diagnosed for pneumonia in relation to x-ray images of healthy lungs.

A visual representation of the ratio of the number of learning data sets to the validation set:

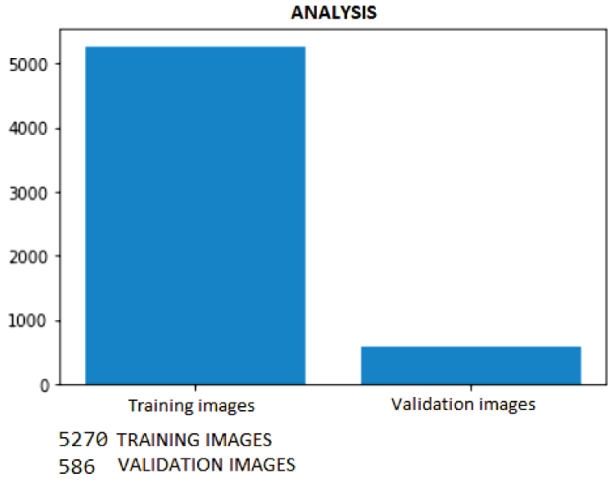

Figure 9 Comparison of validation and training data

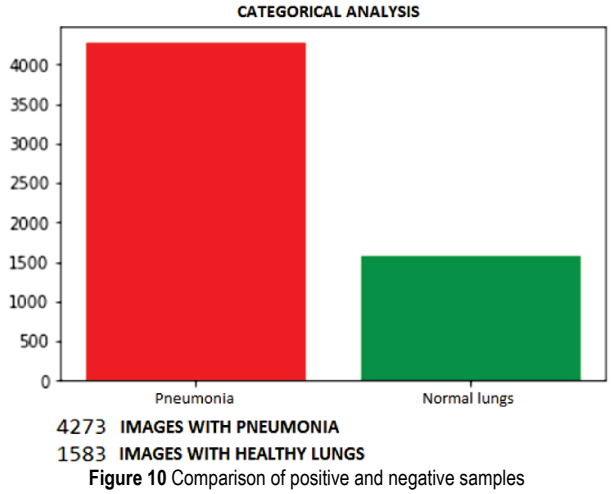

Figure 10 Comparison of positive and negative samples

3.4 Model Architecture

Defining a convolutional neural network architecture is the most important aspect on which the overall success of a project depends. This part of the paper was done using the transfer learning methodology described earlier in the paper, which uses defined architectures available for use in order to have a better model performance. The VGG16 architecture was used in this paper. In this way, the neural network can take advantage of all previously learned principles such as recognizing edges, angles, discoloration, etc., thus shortening learning time and improving model efficiency.

The VGG16 architecture was loaded using a function from the Keras library. Another layer of average pooling layer was added to the defined VGG16 architecture. The last layer of the neural network was changed, which defines the number of classes of possible outputs, in our case 2, that is, pneumonia and normal state of lungs. A softmax activation function has been added to the last layer, which will change the output values depending on the ratio in values ranging from 0 to 1 in order to easily obtain a percentage of the prediction value.

The initial VGG16 architecture and the changes on the last layer are integrated into a single architecture, after which the complete architecture is stored in a variable that will execute the learning process. The functions for calculating the error, the type of optimizer with the learning rate and the values to be monitored during the iterations of the learning process are defined, in our case the prediction accuracy and the error value is included by default. The use of Tensorboard enables a later overview of the learning flow and changes of defined parameters using Tensorboard technology.

After completing all of the above procedures, the model architecture has been successfully implemented and is ready to perform a model training process that may take some time depending on the computer configuration and the amount of learning data.

3.5 Training Process

The process of learning a defined model is the part where the so-called neural network magic takes place. The model will in an iterative way, using the backpropagation algorithm, learn which weight values are best and most effective for producing a model that can detect pneumonia. Once the convolutional neural network architecture has been implemented, the path of the model in which the weighted learning values will be stored is determined. An option is selected that allows only the best model to be saved, to save memory, and to avoid storing weight values at each iteration. The fit function in the domain of Keras technology is called to perform learning process. Images and labels from the learning set and the validation set are sent as parameters, epochs, checkpoints for storing weights, and Tensorboard for later graphical representation of learning curves after learning are determined.

3.6 Model Results

A learning curve that shows the evolution of model accuracy by epochs and the decrease in the amount of error is presented using Tensorboard technology. Graphs were drawn to visualize the accuracy of the model on the learning set across epochs, the error drop on the learning set, and the accuracy percentages and the error drop on the validation set. The tensorboard can be open by typing the tensorboard --logdir = logs / command in the Anaconda command line. The word logs / indicates the directory where graphs are stored. The localhost: 6006 address is then entered in the browser and the Tensorboard window in the browser opens. Graphs related to the results obtained from the learning set:

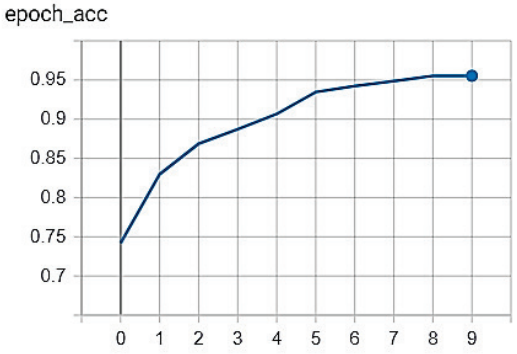

Figure 11 Accuracy curve on learning set

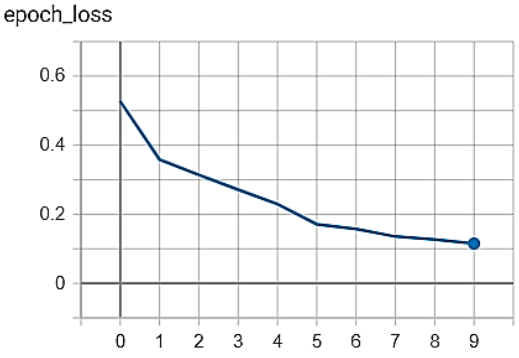

Figure 12 Error curve on learning set

The curves produced from the learning dataset show us an increase in accuracy across all 10 learning epochs, while the error decreases, which is the goal of the model presented. A much more important aspect of the model is the results based on a validation set that has not been used in the learning process to see the model's response to unprecedented data. Graphs from validation set of x-ray image data are shown in the Figs. 13 and 14.

The results on the validation dataset show an overall accuracy of 94%, which is an indicator of the high performance of the pneumonia prediction model. [7]

After the epoch that was one before the last one, it is noticeable that the amount of error at the validation set starts to increase, while at the training set it continues to decrease, which emphasizes the possibility of overfitting. It is necessary to stop the learning process after the 10th epoch, which proves the optimal value of the number of epochs for this type of problem.

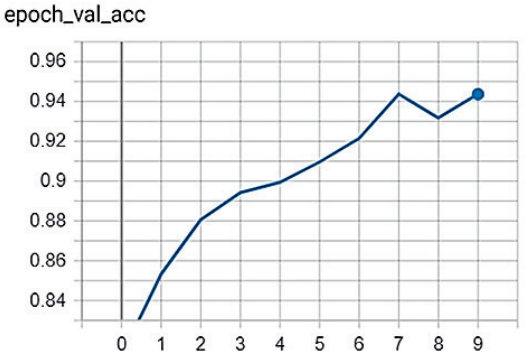

Figure 13 Accuracy curve on validation set

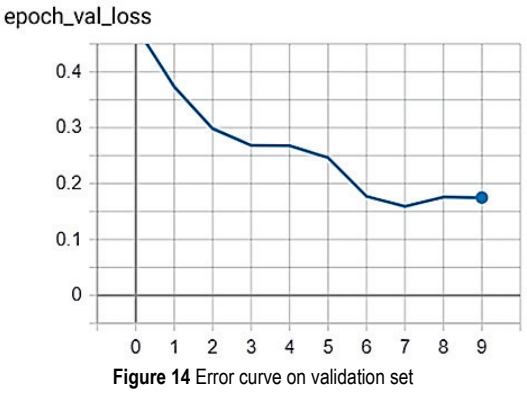

Figure 14 Error curve on validation set

After visualizing the learning process, a confusion matrix was used to display the number of correctly and incorrectly classified images on a validation dataset divided into two classes. The machine learning confusion matrix serves as an instrument for conducting the evaluation of the learned model in the domain of classification problems.

After graphical representation of the confusion matrix, using the Keras evaluate function, the overall accuracy of the model on the validation dataset is 94%. Accuracy is determined on the validation dataset because in the case of evaluation of the learning set, overfitting may occur, and in that way the true accuracy of the model cannot be determined.

The 94% accuracy rate indicates that the model is very well trained on a relatively small dataset of 5000 images. A very important factor in this is transfer learning, in which the architecture of the VGG16 network uses some of the previously acquired knowledge and achieves high accuracy in categorizing pneumonia.

The Fig. 15 shows a visualized confusion matrix created as a product of the steps described above.

By analysing the confusion matrix shown, the high accuracy of the model can be noticed. Of the 429 test RTG images with pneumonia, 415 images showed pneumonia, while only 14 images were misclassified.

In the test dataset, there are significantly fewer images without pneumonia, and out of a total of 157 images, 138 were correctly classified, and for 19 images the model gave incorrect results.

The matrix shows a diagonal shape in darker colours, which is a good sign since the confusion matrix with the strength of its diagonal shows the superiority of the learned model. The total accuracy of the model is below the matrix when all the images are combined together, and it shows the 94% accuracy already indicated.

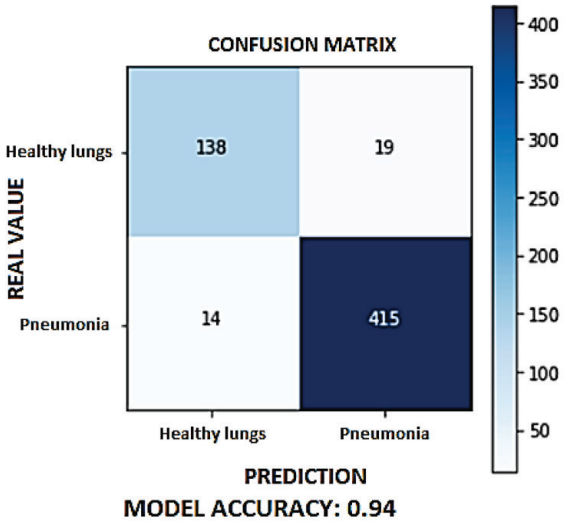

Figure 15 Model confusion matrix

4 CONCLUSION

The aim of this paper was to develop a model of an intelligent system that receives x-ray image of the lung as an input parameter and, based on the processed image, returns the possibility of pneumonia as an output. The implementation of the mentioned functionality was implemented using a transfer learning methodology based on already defined convolution neural network architectures.

In this example, the VGG16 architecture was used, consisting of a total of 16 layers, which greatly contributes to the accuracy of the model using previously learned principles for the classification of image data. After validation of the system, the model shows extremely good results on the validation dataset. For a more complete and qualitative prediction of pneumonia, more data should be available to make the model more representative for decision making and to assist radiologists in making diagnoses.

The model was later integrated with the technologies used to build web applications using the Flask framework. Creating a web application based on the implemented model contributes to the availability of the model, since web applications are available from anywhere at any time, with the requirement that they are connected to the internet. Intuitiveness and ease of use of the application are the main factors for widespread use, which would reduce the number of misdiagnosis of pneumonia.

Acknowledgements

This paper describes the results of research being carried out within the project "Centar održivog razvoja"/"Center of sustainable development", co-financed by the European regional development fund and implemented within Operational Programme Competitiveness and Cohesion 2014 – 2020, based on the call "Investing in Organizational Reform and Infrastructure in the Research, Development and Innovation Sector".

5 REFERENCES

[1] Image dataset, https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia (02.10.2019)
[2] Hrnčić, M. (2019). Design of detector model for RTG imaging, Polytechnic of Međimurje in Čakovec. https://repozitorij.mev.hr/islandora/object/mev%3A1004 (18.09.2019)
[3] Bishop, C. (2007). *Pattern Recognition and Machine Learning*. Springer, pp 738.
[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press, pp 787.
[5] Nielsen, M. (2019). *Neural Networks and Deep Learning*. Online book.
[6] Backpropagation algorithm, https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd (23.09.2019)
[7] Activation functions, https://ieeexplore.ieee.org/document/8404724 (25.09.2019)
[8] Williams, A. (2017). *Convolutional Neural Networks in Python*. CreateSpace Independent Publishing Platform, pp 140.
[9] Transfer learning, https://machinelearningmastery.com/transfer-learning-for-deep-learning/ (29.09.2019.)
[10] Gulli, A. (2017). *Deep Learning with Keras*. Packt Publishing, pp 318.
[11] Tensorboard, https://www.tensorflow.org/tensorboard (02.10.2019)
[12] Confusion matrix, https://ieeexplore.ieee.org/document/7326461 (02.10.2019)

Author's contacts:

Željko KNOK, mr. sc., senior lecturer
Polytechnic of Međimurje in Čakovec,
Bana Josipa Jelačica 22a, 40000 Čakovec, Croatia
zknok@mev.hr

Klaudio PAP, PhD, Prof.
University of Zagreb,
Faculty of Graphic Arts,
Getaldićeva 2, 10000 Zagreb, Croatia
kpap@grf.hr

Marko HRNČIĆ, student
Zagreb University of Applied Sciences
Mlinarska cesta 38, 10000 Zagreb, Croatia
mahrncic@gmail.com