HYBRID MODEL FOR DETECTING LUNG DISEASES

INTRODUCTION

In recent years, Convolutional Neural Networks (CNN) have made great progress across many image recognition tasks. Convolutional layers, pooling layers, fully linked layers, and so on comprise a standard CNN. CNN's feature segmentation and recognition components are the most visible. Convolutional and pooling layers are used to extract features from input photographs, while fully connected layers are used to classify the images. A convolutional, pooling, and fully connected layer series, as well as extra layers such as loss layers, batch normalized layers, and so forth. CNNs are an end-to-end model in which the input images fed into the network are characterized using feature maps and then sorted into one of the many predetermined classes.

Lung disease, in all of its characteristics, is one of the most promising study topics today. With the many medical imaging tasks in hospitals. There are several ways for identifying medical images, but each has a limitation. With medical technology in computer vision, a deep convolutional neural network design has shown excellent outcomes. I presented CNN, which was developed to classify lung diseases. Using X-ray scans, this study employs deep learning to identify whether a person has a lung condition such as lung opacity, Covid-19, or Pneumonia. The project's aim is to predict if a lung has a disease or is healthy based on X-ray pictures using TensorFlow, Keras, and Scikit Learn.

RELATED WORK

Bharti et al. [1] propose VDSNet, a new hybrid deep learning architecture for identifying lung illness using X-rays. The model is applied to a set of NIH chest X-ray imaging data from the Kaggle library. VDSNet has the maximum validation precision of 73 percent for entire data sets, whereas Vanilla Gray, Vanilla RGB, Hybrid CNN VGG, Basic CapsNet, and Modified CapsNet have accuracy values of 67.8 percent, 69 percent, 69.5 percent, and 60.5 percent, respectively. The VDSNet validation accuracy value is 73 percent, which is higher than the sample data set precision value of 70.8 percent. VDSNet, on the other hand, needs 31 seconds to train on a whole data set. This takes substantially longer than the sample data set's 19 seconds.

Trusculescu et al [2] explain the evolution of deep learning algorithms and their applications in medicine, particularly in the diagnosis of PID. It gave implementation possibilities that lead to clinically essential everyday operations for early diagnosis of PID, in addition to the obstacles. The next frontier in the early diagnosis of IPF is the creation of CAD that can be utilized at any computer station and is accessible to non-academic centers.

Sriporn et al. [3] used a deep learning strategy to identify indications of lung infection using the DenseNet121 network vs other network models such as ResNet50 and MobileNet. The goal of this work was to evaluate the efficiencies of the three most common MobileNet CNN models, Resnet50 and Densnet121, as well as predict lung illness utilizing Mish and seven optimization approaches. Its purpose

was to boost the efficiency of these CNN models. Between the standard CNN model and the CNN model with Mish, there are seven optimization strategies for predicting lung illnesses.

Alqudah et al. [4] focused their research to adopting artificial intelligence methods to detect COVID-19 early in chest X-ray images. Different hybrid models – each comprised of deep feature extraction and classification approaches – are employed to assist clinicians in the detection of COVID-19. A convolutional neural network is used to extract graphical properties from chest X-ray images in the hybrid model implementations (CNN). Several techniques, including CNN, support vector machine (SVM), and random forest (RF), are utilized to identify pictures as COVID-19 or non-COVID-19 to get the best recognition performance. The two most essential retrieved characteristics are utilized for training and parameter testing. According to the results of the constructed models, CNN outperforms other classifiers with a testing accuracy of 95.2%.

Polsinelli et al. [5] present a simple Convolutional Neural Network (CNN) architecture based on the SqueezeNet model for classifying COVID-19 CT images from other community-acquired pneumonia and/or healthy CT images. With a 3.2 percent increase in the first dataset arrangement and a 2.1 percent improvement in the second dataset arrangement, the design achieves an accuracy of 85.03 percent. Due to the small size, the obtained benefit can be highly beneficial in medical diagnostics, notably in the Covid-19 condition. Furthermore, the average classification time on a high-end workstation, 1.25 s, is competitive with the 13.41 s of more complex CNN architectures that need pre-processing. The recommended CNN can be executed in 7.81 seconds on a mid-range laptop without GPU acceleration, which is impossible for techniques that need GPU acceleration. The method's performance can be further improved by applying efficient pre-processing methods that do not require GPU acceleration.

Dansana et al. [6] Scanning X-ray and computed tomography (CT) images is one of the most promising areas of study; it may assist in sick detection and early diagnosis, and it delivers both quick and accurate findings. This study used convolution neural networks for binary classification pneumonia-based conversion of VGG-19, Inception V2, and decision tree models on a dataset of 360 X-ray and CT scan images. The fine-tuned version VGG-19, Inception V2, and decision tree models all perform exceptionally well, with a rate of development in training and validation accuracy (91%) that beats the Inception V2 (78%) and decision tree (60 %) models.

DATA

The dataset is derived from the Kaggle. A team of researchers from Qatar University, Doha, Qatar, and the university of Dhaka, Bangladesh along with their collaborators from Pakistan and Malaysia in collaboration with medical doctors have created a database of chest X-ray images for Covid-19 positive cases along with normal, and pneumonia images. X-ray images are used to analyze the health of the patients' lungs. There was total 3,616 Covid-19 Images, 6,012 Lung opacity images, 10,192 normal images and 1,345 viral Pneumonia images. The total number of images in this dataset are 21,165. In this task, my proposed methodology would be evaluated on this dataset.

METHODOLOGY

I have used Jupyter Notebook and imported some important libraries as such; Convolutional Neural Network (CNN) & Keras from TensorFlow, scikit learn, cv2, matplotlib, numpy and pandas. As I already mentioned, images of different classes are in different folders. I have created a common function to visualize them separately. The images are then analyzed to get a clear idea about whether the lungs are affected or not with any disease. Below in figure 1 shows X-ray images that I used to train the neural network model by setting the target variables of lung diseases as; Lung-opacity, Covid-19, pneumonia and a normal person lungs.

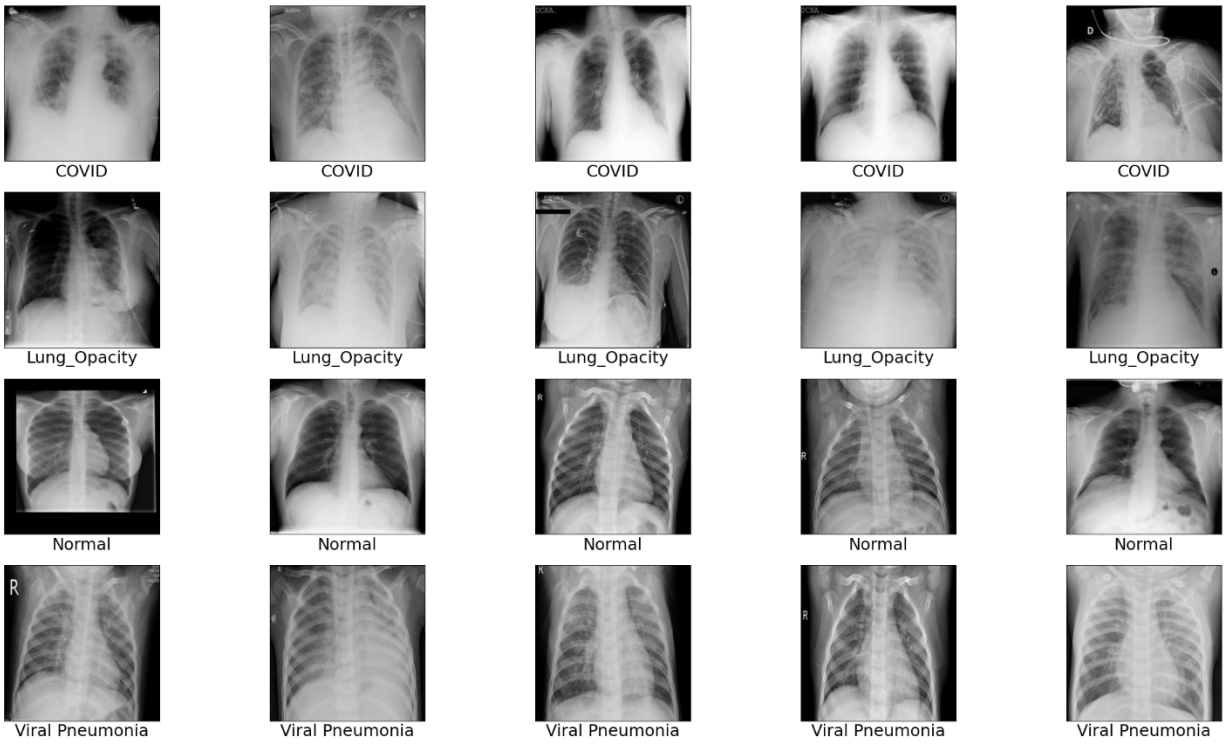

Figure 1: *Normal vs Different Lung diseases*

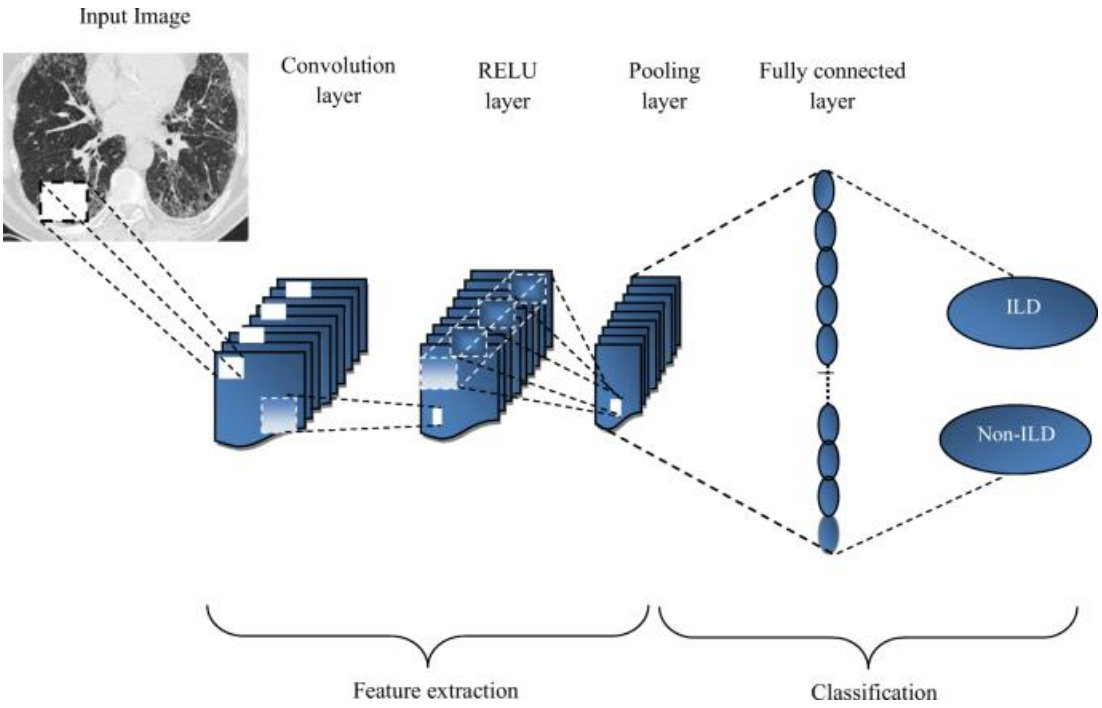

Figure 2: *Convolutional neural network architecture*

Reference page: *https://link.springer.com/article/10.1007/s00330-020-06986-4*

- **Preprocessing the image data:**

Every machine learning model needs the data as numbers to make the prediction. So,To model them, I transformed the dataset into an array of numbers. I wrote a function that preprocesses images to arrays using the Keras image to array algorithm, which turns images into numerical arrays. Later, I used LabelBinarizer to transform the dataset's category target label to a number.

Scaling is an important step in any data-processing for the model to perform better.The dataset is subjected to minmax scaling in order to be converted into a range of 0 to 1. To carry them out, the array is split by 255, which is the max value that any image array can have.

It is necessary to test the model's performance using the hidden dataset (test/validation data). So, with the aid of the Sklearn tool, I divided the dataset into train and test sets. I've put aside 20% of the data as a test set.

To apply TensorFlow, my data has to be in tensor format. So, I changed all the data into tensors. To get the same result, I applied TensorFlow's Image Data Generator function. To acquire the original images as tensors, I set all the parameters at their default values.

❖ CNN Model:

```
model_custom = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model_custom.add(Conv2D(32, (3, 3), padding="same",activation="relu",input_shape=inputShape, strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(3, 3)))

model_custom.add(Conv2D(64, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(64, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(2, 2)))

model_custom.add(Conv2D(128, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(128, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(2, 2)))

model_custom.add(Conv2D(512, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(512, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(512, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(2, 2)))

model_custom.add(Flatten())
model_custom.add(Dense(1024, activation="relu"))
model_custom.add(Dropout(0.5))
model_custom.add(Dense(4,activation="softmax" ))
```

```
model_custom.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 256, 256, 32)      896
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 85, 85, 64)        18496
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 85, 85, 64)        36928
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 42, 42, 64)        0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 42, 42, 128)       73856
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 42, 42, 128)       147584
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 21, 21, 128)       0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 21, 21, 512)       590336
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 21, 21, 512)       2359808
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 21, 21, 512)       2359808
_________________________________________________________________
```

```
max_pooling2d_13 (MaxPooling (None, 10, 10, 512)       0
_________________________________________________________________
flatten_2 (Flatten)          (None, 51200)             0
_________________________________________________________________
dense_7 (Dense)              (None, 1024)              52429824
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 4)                 4100
=================================================================
Total params: 58,021,636
Trainable params: 58,021,636
Non-trainable params: 0
```

After various regressive modifications and analyzing the performance of each model by changing various hyper parameters, I was able to train a better performing model. The structure of the model is displayed below.

→ 1 x Convolution 2D layer with 32 filter channels of 3x3 kernel and same padding

→ 1 x Maxpool2D layer of 2x2 pool size

→ 2 x Convolution 2D layer with 64 filter channels of 3x3 kernel and same padding

→ 1 x Maxpool2D layer of 2x2 pool size

→ 2 x Convolution 2D layer with 128 filter channels of 3x3 kernel and same padding

→ 1 x Maxpool2D layer of 2x2 pool size

→ 3 x Convolution 2D layer with 512 filter channels of 3x3 kernel and same padding

→ 1 x Maxpool2D layer of 2x2 pool size

I applied relu (Rectified Linear Unit) activation to each convolution layers so that all the negative values are not passed to the next layer. After creating all the convolution, I pass the data to the dense layer so for that I flatten the vectors which comes out of the convolutions and added dense layers.

→ Flatten the above maxpool result

→ 1 x Dense layer of 1024 units; It extracts the parameters

→ 1 x Dropout of 0.5

→ 1 x Dense Softmax layer of 4 units

I got total trainable params in model summary was 58,021,636. Same Adam optimizer was used on this model too, to train the model.

RESULT AND DISCUSSIONS

After 100 epochs, model's accuracy has improved and got the accuracy of 94.7% on train set and 84.03% on the test set. The increase of accuracy and decrease of loss for each epoch can be seen in the below graph.

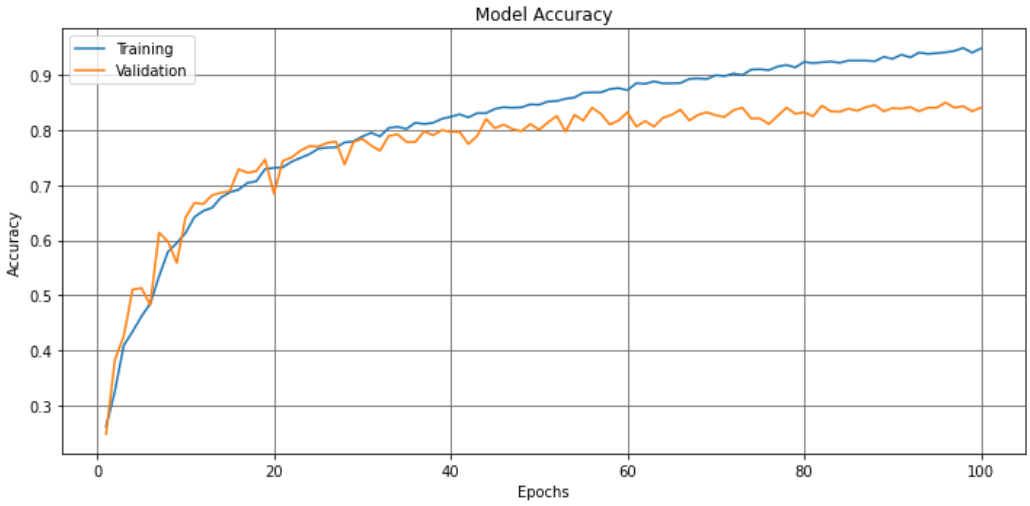

Figure 3: *Model Accuracy*

After looking at figure 3 above we can see that the accuracy on validation dataset does not change after 30 epochs. Hence, we can say that this model performs good after only 30 epochs which is good for training the model.

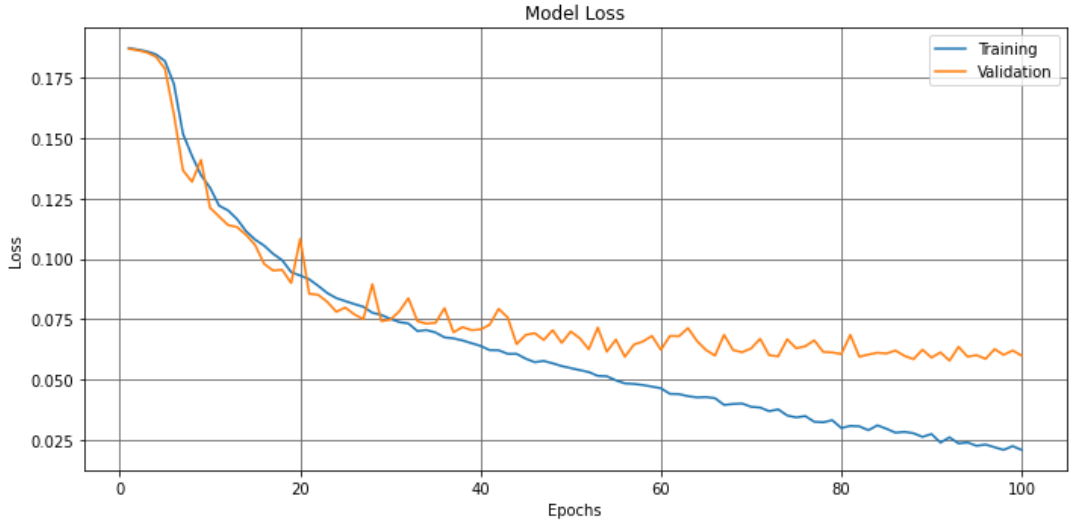

Figure 4: *Model Loss*

After looking at figure 4 above we can see that the loss of the model is constant after 30 epochs like figure 3. Also, you can see that the loss value is very less, hence we can state that this model is good model for classifying the images accordingly.

CONCLUSION AND FUTURE WORK

After training template model and CNN models. I was able to train a better performing model with good accuracy. After looking at the figure 3 and 4 we can say that this model can be optimize after only 30 epochs with a good accuracy and very less model loss. The accuracy of the model was 94.7% on train set and 84.03% on the test set. It is also taking a less time.

From this project, I was able to understand and implement various steps in image processing. I got a deep knowledge about the convolutional neural network. Now with transfer learning, the weights of this trained model can be used to implement other use cases too. I can train model with CT scan images dataset, can provide various options in future to detect other problems of body using various medical reports. My future work would be developing a web app or android app. So, doctors or in medical institution it will be easy to detect the lung diseases using the app. Also, with this app, X-ray can be tested to find whether the patient have lung disease or not.

RFERENCES

1. Bharati, Subrato, Prajoy Podder, and M. Rubaiyat Hossain Mondal. "Hybrid deep learning for detecting lung diseases from X-ray images." *Informatics in Medicine Unlocked* 20 (2020): 100391.
2. Trusculescu, Ana Adriana, Diana Manolescu, Emanuela Tudorache, and Cristian Oancea. "Deep learning in interstitial lung disease—how long until daily practice." *European radiology* (2020): 1-8.
3. Sriporn, Krit, Cheng-Fa Tsai, Chia-En Tsai, and Paohsi Wang. "Analyzing lung disease using highly effective deep learning techniques." In *Healthcare*, vol. 8, no. 2, p. 107. Multidisciplinary Digital Publishing Institute, 2020.
4. Alqudah, Ali Mohammad, Shoroq Qazan, Hiam Alquran, Isam Abu Qasmieh, and Amin Alqudah. "Covid-19 detection from x-ray images using different artificial intelligence hybrid models." *Jordan Journal of Electrical Engineering* 6, no. 2 (2020): 168-178.
5. Polsinelli, Matteo, Luigi Cinque, and Giuseppe Placidi. "A light CNN for detecting COVID-19 from CT scans of the chest." *Pattern recognition letters* 140 (2020): 95-100.
6. Dansana, Debabrata, Raghvendra Kumar, Aishik Bhattacharjee, D. Jude Hemanth, Deepak Gupta, Ashish Khanna, and Oscar Castillo. "Early diagnosis of COVID-19-affected patients based on X-ray and computed tomography images using deep learning algorithm." *Soft Computing* (2020): 1-9.
7. Masud, Mehedi, Anupam Kumar Bairagi, Abdullah-Al Nahid, Niloy Sikder, Saeed Rubaiee, Anas Ahmed, and Divya Anand. "A Pneumonia Diagnosis Scheme Based on Hybrid Features Extracted from Chest Radiographs Using an Ensemble Learning Algorithm." *Journal of Healthcare Engineering* 2021 (2021).
8. Varshni, Dimpy, Kartik Thakral, Lucky Agarwal, Rahul Nijhawan, and Ankush Mittal. "Pneumonia detection using CNN based feature extraction." In *2019 IEEE international*

conference on electrical, computer and communication technologies (ICECCT), pp. 1-7. IEEE, 2019.

9. Sriporn, Krit, Cheng-Fa Tsai, Chia-En Tsai, and Paohsi Wang. "Analyzing lung disease using highly effective deep learning techniques." In *Healthcare*, vol. 8, no. 2, p. 107. Multidisciplinary Digital Publishing Institute, 2020.
10. Momeny, Mohammad, Ali Asghar Neshat, Mohammad Arafat Hussain, Solmaz Kia, Mahmoud Marhamati, Ahmad Jahanbakhshi, and Ghassan Hamarneh. "Learning-to-augment strategy using noisy and denoised data: Improving generalizability of deep CNN for the detection of COVID-19 in X-ray images." *Computers in Biology and Medicine* 136 (2021): 104704.
11. M. E. H. Chowdhury et al., "Can AI Help in Screening Viral and COVID-19 Pneumonia?," in IEEE Access, vol. 8, pp. 132665-132676, 2020, doi: 10.1109/ACCESS.2020.3010287.
12. Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S.B.A., Islam, M.T., Maadeed, S.A., Zughaier, S.M., Khan, M.S. and Chowdhury, M.E., 2020. Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images.
13. https://www.kaggle.com/gpiosenka/f1-test-score-95/notebook
14. https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c
15. Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". Mathematics and Computers in Simulation. Elsevier BV. 177: 232–243. doi:10.1016/j.matcom.2020.04.031. ISSN 0378-4754. Convolutional neural networks are a promising tool for solving the problem of pattern recognition.
16. Govindaswamy A., Montague E., Raicu DS., Furst JD., "CNN as a feature extractor in gaze recognition", Independent Study, College of Computing and Digital Media, DePaul University, 2020.
17. https://link.springer.com/article/10.1007/s12539-020-00403-6