Determining Pneumonia through Chest X-Rays using Convolutional Neural Network

Rishika Chhabria
AI Tech Systems
(www.ai-techsystems.com)
Mumbai , India
rishika.chhabria@gmail.com

***Abstract*—Pneumonia is an infection that causes inflation in the lungs due to virus, bacteria or other germs. Chest X-Ray is used to diagnose this infection. The automatic detection of pneumonia using chest x-ray has become an economical medical diagnostic technique. This paper evaluates an algorithm which can be used to classify whether a person has pneumonia or not using their chest x-ray images. Convolutional Neural Network is used to classify the x-ray images into two categories (Pneumonia/Healthy) using Classification model and Keras library.**

Keywords—Pneumonia, Chest X-Ray, Convolutional Neural Network (CNN), Classification, Keras .

I. INTRODUCTION

Pneumonia [1] is an inflammatory disease in the lungs in the alveoli .When the lungs are affected by viruses or bacteria, the pus accumulates in the alveoli leading to Pneumonia. Pneumonia is the single largest infectious cause of death worldwide. It accounts for 15% of all deaths of children worldwide [2]. Physicians use chest x-ray image to diagnose or monitor the treatment of Pneumonia. However, Computer-aided Diagnostic tools (CADx) help in easier diagnoses of the disease. Convolutional Neural Network models help these tools in diagnosing by extracting image features for visual recognition. A neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN consist of a series of layers such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves back propagation in order to more accurately weight the end product [3].

This paper evaluates the classification model by using Rectified Linear Unit (ReLU) activation function and Max pooling. The paper presents a detailed evaluation steps performed to test the accuracy and F1 score of the test set. In this paper steps involved right from building layers, hidden layers as well as output layers to obtain the accuracy score (test set) of 85.08%, recall score of 93.84% and F1 score of 75.23 % is presented.

II. METHODOLOGY

A. Dataset

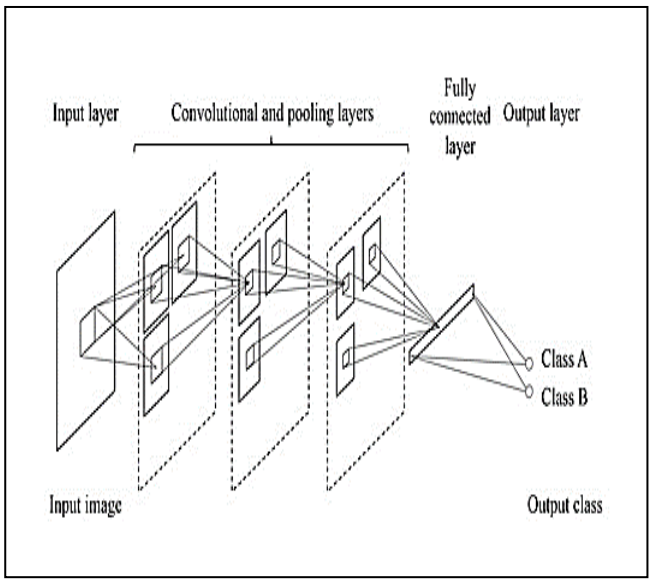
The dataset was taken from Kaggle [4]. It was divided into 3 folders (Test, Train, Val). Each of these folders was divided into sub-folders (Normal/Pneumonia). Test set contained 624 images, Train set contained 5216 images and Val set contained 16 images. Keras open-source deep-learning framework along with Tensorflow backend was deployed to build and train Convolutional Neural Network model.

B. Model

The model proposed was Convolutional Neural Network (CNN) [5]. It is a special type of neural network that consists of one or more convolutional layers, a sub-sampling layer which is followed by one or more fully-connected layers. Fig 1. shows the overall architecture of CNN.

Convolutional Layer [5] extracts the different features of input. The first convolution layer extracts low-level features like edges, lines, and corners. Higher-level layers extract higher-level features. The pooling layer reduces the resolution of the features. There are two ways to do pooling: max pooling and average pooling. In max pooling, maximum pixel value of the batch is selected. In average pooling, average pixel value of the batch is selected. This model uses a specific activation function Rectified Linear Unit having the following equation: $y = \max(x, 0)$ so that input and output size of the layer is same. The output of the convolution and max pooling are assembled into 2D planes called feature maps. Fully connected layers are the final layers of a CNN. In case of these layers, all the elements of all the features of the previous layer get used in the calculation of each element of each output feature. Each input image will pass through a series of layers giving probabilistic values between 0 and 1.

Fig 1. CNN Architecture [6]



III. IMPLMENTATION

In this model [7], first layer used was the Sequential Layer to let the model know the input shape. Following it ,4 Conv2D layers were used. Convo2D is a 2D Convolutional Layer. The kernel generated by this layer is convolved with input layer to produce outputs. The dimensions of each layer were given as 32. The activation function was ReLU. Along with 4 Convo2D layers, 4 MaxPooling2D layers were also used. The parameters given to it was a pool size tuple (2,2). Pool size is the size of maximum pooling windows. The output of convolutional layers and max pooling layers combine to form feature maps. Each layer takes the preceding layer's output as input and passes its output as input to the succeeding layers.

The core layers used in the model are Flatten and Dense. Flatten layer is a connection layer between Dense and Conv2D layers. Flatten layer is used to flatten the data and convert it into a 1D linear vector array. It is used only when the data is to be passed through the dense layer. The following figure Fig 2. shows the layer arrangements.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 32)	0
conv2d_3 (Conv2D)	(None, 4, 4, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 2, 2, 32)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 128)	16512
dense_1 (Dense)	(None, 1)	129

Fig 2. Layers of the model.

The classifier is then compiled using metrics as 'accuracy', loss as 'binary cross entropy' and optimizer as 'adam' to configure the learning process. The metrics accuracy determines the accuracy of the model by calculating the number of correct predictions out of the total predictions. Binary cross entropy is a loss function which determines the result using binary decisions. Optimizers were used to minimize the loss functions by updating the weight parameters.

The model was then trained using batch size as 32. It was fitted using the validation data as test set and a total of 10 epochs. Every epoch trained 163 images. The number of images were kept less in order to reduce the processing time. An epoch is when the entire dataset is passed, but to reduce the processing time it is divided into smaller batch sizes. Using only 1 epoch leads to underfitting. As the number of epochs are increased, the weights are changed more often and hence the curve improves to an overfitting curve. When the data is passed through each layer, it generates some error along with the outputs generated. To minimize this loss or error, back-propagation method is used. This information about the error is then backpropagated by the network to adjust the weights and minimize the error. This process is followed after every output is generated.

Despite of passing 163 images in 1 epoch and processing 10 such records, the recall, precision, F1 score and accuracy obtained was decent enough. Fig 3 shows the epochs.

```
Epoch 1/10
163/163 [==============================] - 82s 502ms/step - loss:
0.5272 - acc: 0.8077 - val_loss: 0.4412 - val_acc: 0.7949
Epoch 2/10
163/163 [==============================] - 77s 470ms/step - loss:
0.2853 - acc: 0.8834 - val_loss: 0.7934 - val_acc: 0.7340
Epoch 3/10
163/163 [==============================] - 77s 471ms/step - loss:
0.2286 - acc: 0.9072 - val_loss: 0.4870 - val_acc: 0.7981
Epoch 4/10
163/163 [==============================] - 77s 474ms/step - loss:
0.2297 - acc: 0.9126 - val_loss: 0.3974 - val_acc: 0.8285
Epoch 5/10
163/163 [==============================] - 77s 470ms/step - loss:
0.2167 - acc: 0.9197 - val_loss: 0.3794 - val_acc: 0.8397
Epoch 6/10
163/163 [==============================] - 77s 470ms/step - loss:
0.2019 - acc: 0.9212 - val_loss: 0.4886 - val_acc: 0.8077
Epoch 7/10
163/163 [==============================] - 77s 470ms/step - loss:
0.1863 - acc: 0.9277 - val_loss: 0.6154 - val_acc: 0.7740
Epoch 8/10
163/163 [==============================] - 77s 471ms/step - loss:
0.1827 - acc: 0.9308 - val_loss: 0.3887 - val_acc: 0.8429
Epoch 9/10
163/163 [==============================] - 77s 471ms/step - loss:
0.1883 - acc: 0.9270 - val_loss: 0.6708 - val_acc: 0.7532
```

Fig 3.Epochs

IV. RESULTS

To evaluate the effectiveness of the model, experiments was conducted 4 to 5 times. Parameters were altered, features such as batch size, epochs were changed to validate the effectiveness. The results obtained indicate that larger the size of the image, smaller is the accuracy. Larger images take more training time and computation cost. On the other hand, smaller images get trained early. My analysis shows that the accuracy increased by increasing the number of filters in the convolutional layers. The test set accuracy was 85.08%. Recall and F1 score was 93.84 % and 75.23 % respectively. The confusion matrix is tabulated in table 1 and the results are tabulated in Table 2.

TABLE 1. CONFUSION MATRIX

17	217
24	366

TABLE 2.CLASSIFICATION REPORT

	Precision	Recall	F1 Score
Normal	0.41	0.07	0.12
Pneumonia	0.63	0.94	0.75

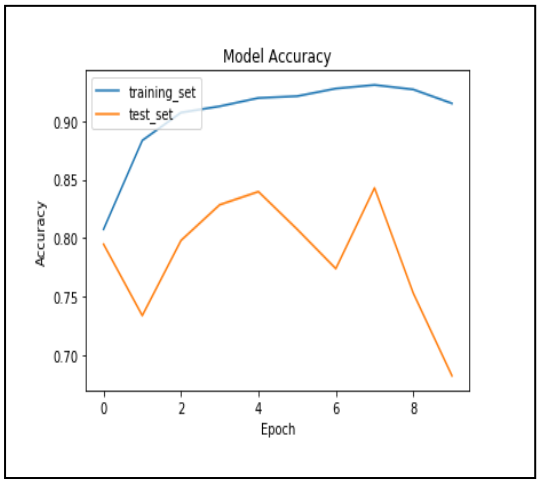


Fig 4. Accuracy Graph

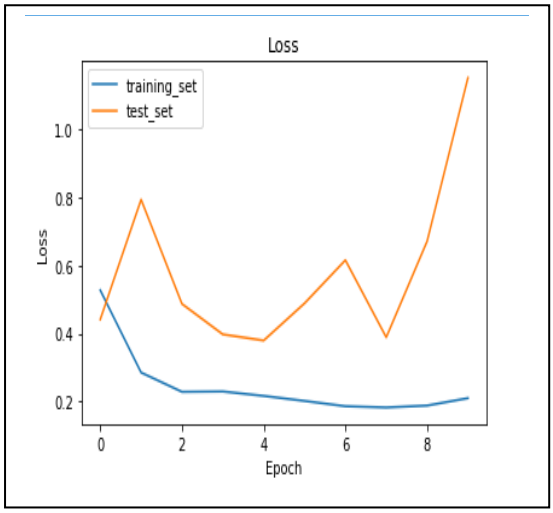


Fig 5. Loss Graph.

V. CONCLUSION

This report presents the model evaluated and validated to produce the results using deep learning. Convolutional Neural Network is a part of deep learning. Determining whether a person has pneumonia or not with the help of deep learning techniques is a demanding process because of variable size of input images.

REFERENCES

[1] http://www.bumc.bu.edu/pneumonia/background/what/

[2] https://www.who.int/news-room/fact-sheets/detail/pneumonia

[3] https://en.wikipedia.org/wiki/Convolutional_neural_network

[4] https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

[5] https://ip.cadence.com/uploads/901/cnn_wp-pdf

[6] https://meritis.fr/wp-content/uploads/2019/01/4_CNN.jpg

[7] https://keras.io