Pneumonia Detection from Chest Xray Image

Md Rakibul Hasan Talukder
Mridul Banik
December 14, 2021
Final Report, CS 545 Machine Learning

Table of Contents

Introduction

Machine Learning allows us to understand dataset without explicitly writing conditions. Different architectures are used to train different type of data type. Convolutional Neural Network is one of it's kind which is used to train on image dataset. We think convolutional neural network is a great architecture that is simple but powerful. The evolution of this architecture also attracted us to work with CNN. There are a lot of application of CNN architecture. It can be used in medical domain, face recognition, landmark recognition, traffic image, satellite image and so on.

We think applying computation in medical domain could be very helpful for mankind. We found out that architecture of CNN can be a great way of solving medical imaging problem.

We decided to work on X-Ray image to classify whether this is pneumonia or normal. This is a great problem domain. Figuring out pneumonia from X-Ray image require and expert which might not be available in rural areas of different country. This system we think, can be vastly useful to inform the patient or doctors regarding state of pneumonia within a second which does not require any human intervention.

Our plan is to experiment with different architecture of CNN to find out the best fit model which gives good accuracy across the training, validation and test set.

Methodology

Data Processing

The Dataset we have collected from Kaggle site [1] comprised of total 5856 images. This dataset used in this project did not include any case of viral and bacterial co-infection. All chest X-ray images were taken during the routine clinical care of the patients. Two expert physicians then graded the diagnoses for the images before being cleared for training the AI system.

We have 3 sets of this dataset. A train set which contains 5216 images, a validation set which contains 624 images and test set with 16 images. The train contained 1341 Normal chest X-ray image and 3875 "Pneumonia" images. In validation set we had 234 "Normal" images and 390 "Pneumonia" images. In test set we had 8 "Normal" and 8 "Pneumonia" images.

We have loaded our dataset in Imagenerator. In train generator we augmented images by number of factors. We rescaled our image by 1/255. The shear rage was 0.2. The Zoom range was 0.2 and kept horizontal_flip value True. For validation and test generator we only rescaled images by 1/255.

Experiments and Results

For classification we used bunch of pretrained networks and couple of custom networks. We used 10 different pretrained model and used their weights in Chest X-ray dataset. Among this Ten, six models were also experimented with dropout rate of 0.4 and result was compared. Apart from these thirteen models, we have run two custom CNN of 7 layers and 17 layers. All these models were compiled with the loss function of "Categorical Cross Entropy". Adam was used as the optimizer function for all models. Now for each model we are going to describe the model briefly, provide an architectural view on the experiment, show the training result and evaluate the models on validation and testing datasets.

VGG 16

VGG16 is widely used CNN used for ImageNet, a large visual database project used in visual object recognition. he VGG16 Architecture was developed and introduced by Karen Simonyan and Andrew Zisserman from the University of Oxford, in the year 2014, through their article "Very Deep Convolutional Networks for Large-Scale Image Recognition." 'VGG' is the abbreviation for Visual Geometry Group, which is a group of researchers at the University of Oxford who developed this architecture, and '16' implies that this architecture has 16 layers [2].

After using the pre-trained model of VGG16 in this model we used 3 more layers. We first flattened the output of pretrained model by a flatted layer then added two dense layers with 128 and 2 units respectively. With these 3 more layers our model had 3,211,650 Trainable parameters and 14,714,688 Non-trainable parameters.

We used 100 steps per epochs and 15 epochs for the model training. On 15 th epoch, we found our training accuracy 0.98 and validation accuracy 0.8670. Our training loss was 0.0534 whereas the validation loss was 0.5456. From the graph we can see, the training was not smooth it has few ups and downs in both accuracy and loss graph. After 10th epoch the validation accuracy started to go down and validation loss started to increase even though the training accuracy was increasing, and training loss was decreasing.

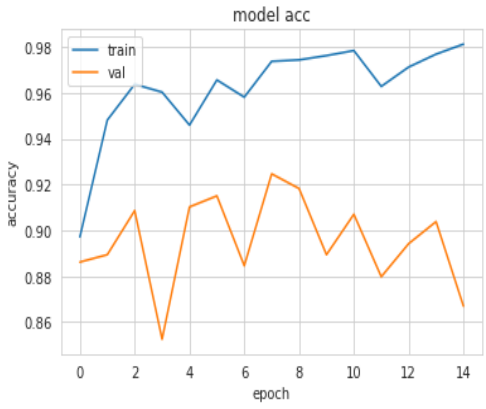


Figure 1 VGG-16

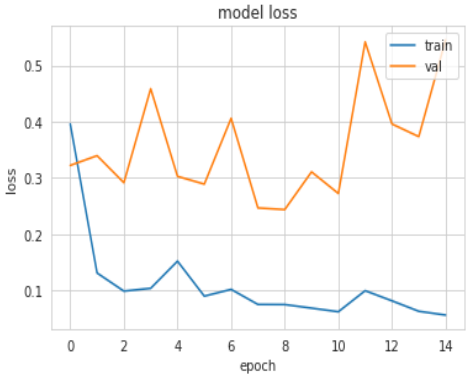


Figure 2 VGG-16

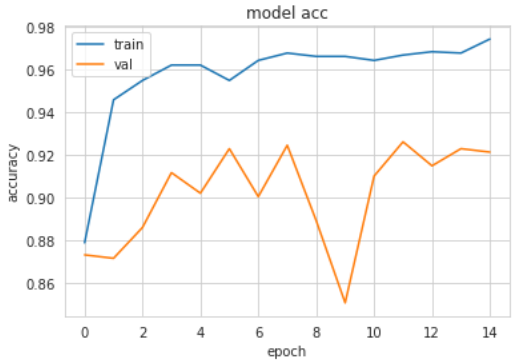


Figure 3 VGG-16 - After Dropout

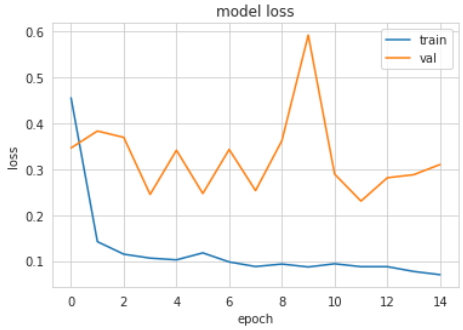


Figure 4 VGG-16- After Dropout

Validation set evaluation

We evaluated our model in the validation which provided us 0.83 precision. However, the model provided a good recall value which is 0.99, The f1 score is 0.9.

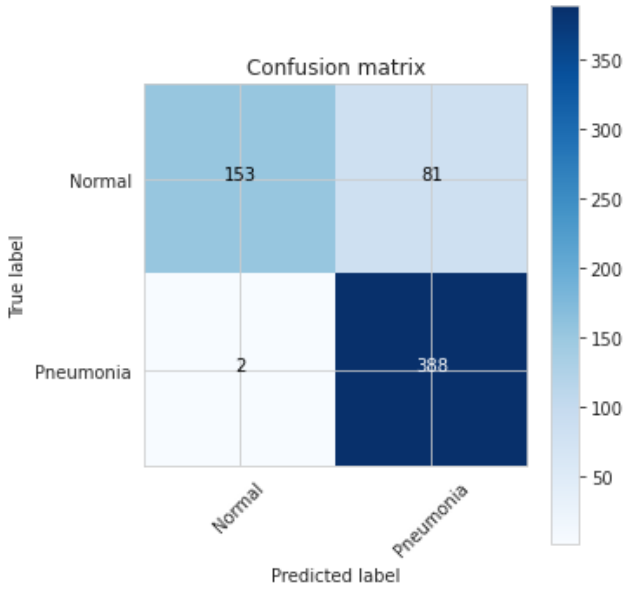


Figure 5 Vgg16 - Validation set

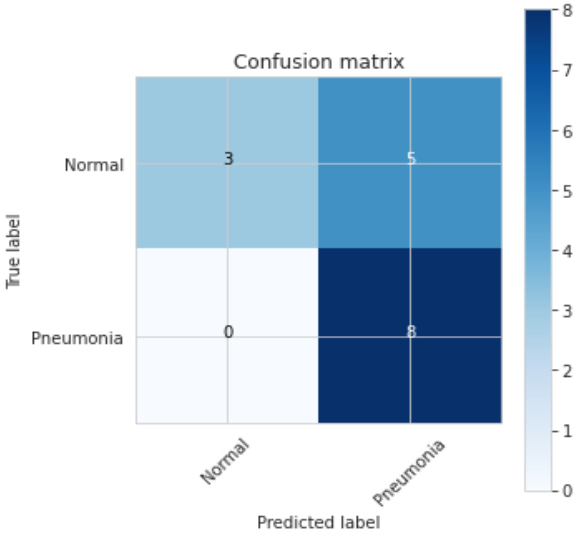


Figure 6 Vgg16 - Test set

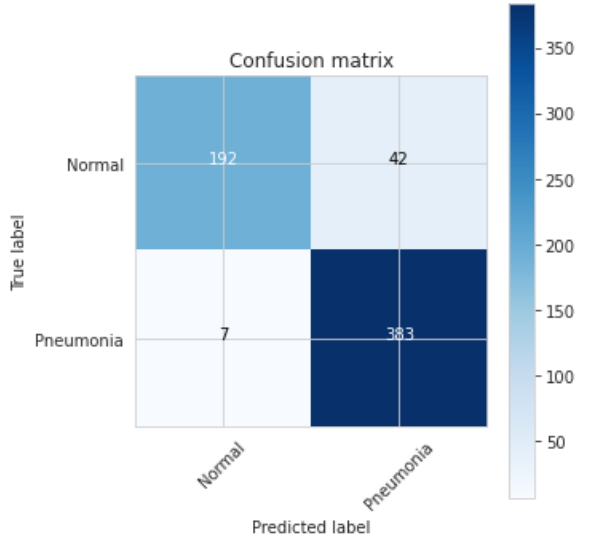


Figure 7 Vgg-16 - After Dropout - Validation Set

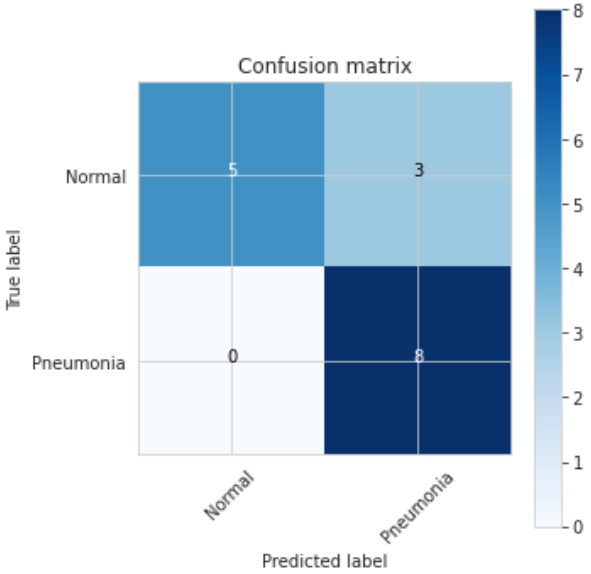


Figure 8 Vgg-16 - After Dropout - Test Set

Test set Evaluation

In the test set, the precision was a bit low with only 0.62. The recall score is 1.00 and the f1 was 0.76.

VGG 16 After Dropout

As it seemed, the model was overfitting, we added a dropout layer with a 0.4 rate to minimize the overfitting. We added the layer after pre-trained model layer. This change provided us with better accuracy. The training accuracy became 0.95 and validation accuracy became 0.91. The accuracy graph seemed to increase and loss seemed to decrease which directed that the accuracy could raise even more with higher epochs.

Validation set evaluation

In validation set the precision score increased as compared to the same model with the dropout layer. The precision score have become 0.90. However, the recall score decreased to 0.98. The f1 score became 0.94.

Test set evaluation

In test set we can see the precision score is 0.73 and recall is 1.00. The f1 score is also 0.84 which made the model a good at predicting true positive classes.

VGG19

VGG19 is a variant of VGG model which in short consists of 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer). There are other variants of VGG like VGG11, VGG16 and others. VGG19 has 19.6 billion FLOPs [1].

After using the pre-trained model of VGG19 in this model we used 3 more layers. We first flattened the output of pretrained model by a flatted layer then added two dense layers with 128 and 2 units respectively. With these 3 more layers our model had 3,211,650 Trainable parameters and 20,024,384. In VGG19 we also used 15 epochs in model training and 100 steps pers epochs. The training accuracy on 15th epoch 0.96 but but the validation accuracy is 0.89. Our training loss was 0.0882 whereas the validation loss was 0.3316. For VGG19 also, the training was not smooth it has few ups and downs in both accuracy and loss graph.

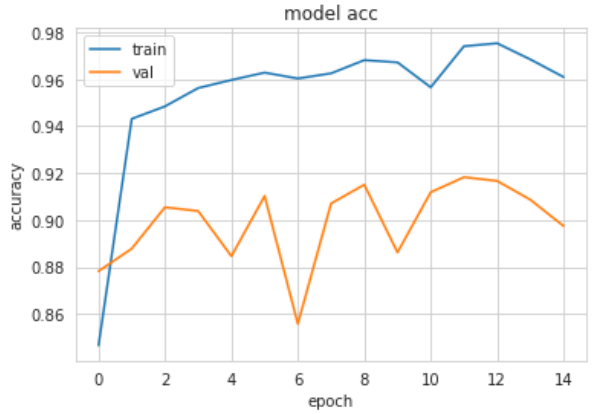


Figure 9 Vgg-19

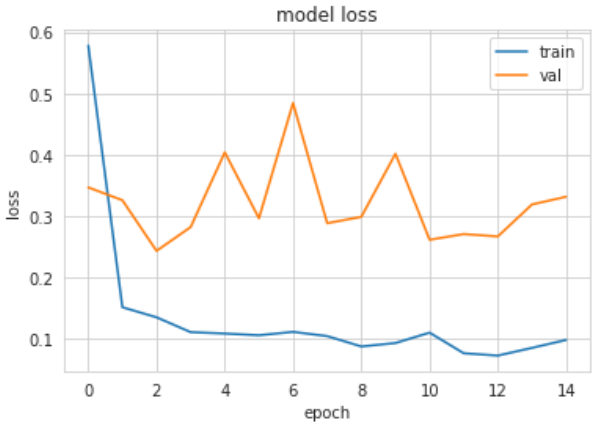


Figure 10 Vgg-19

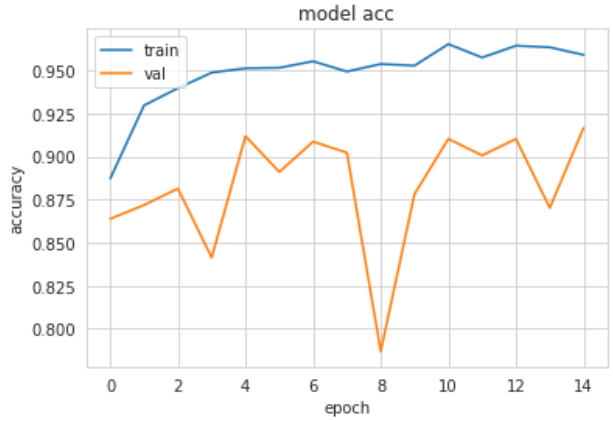


Figure 11 Vgg-19 - After Dropout

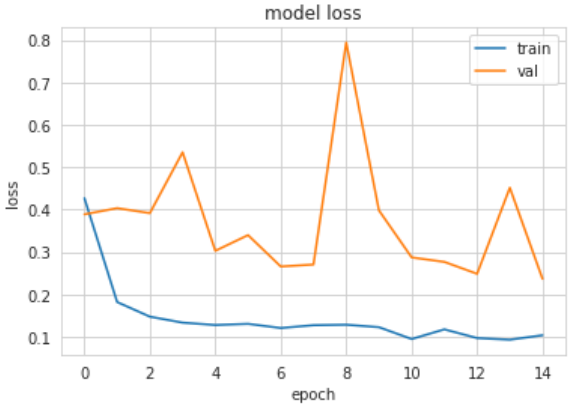


Figure 12 Vgg-19 - After Dropout

Validation set evaluation

In the evaluation process of validation set, the model got 0.87 score in precision. The recall score was good with 0.98. The f1 score is 0.92.

Test set evaluation

In test set we can see the precision score is 0.89 and recall score is 1.0. The f1 score is also 0.94 which can be considered as good model if the context is to correctly identify true positive classes.

VGG19 After Dropout

This model also seemed to be overfitting, so, we added a dropout layer with a 0.4 rate to minimize the overfitting. We added the layer after pre-trained model layer. This change provided us with better accuracy. The training accuracy became 0.9585 and validation accuracy became 0.9167.

Validation set evaluation

In the evaluation process of validation set, the model got 0.93 score in precision. The recall score was good with 0.94. The f1 score is 0.93 which make the model a very balanced one.

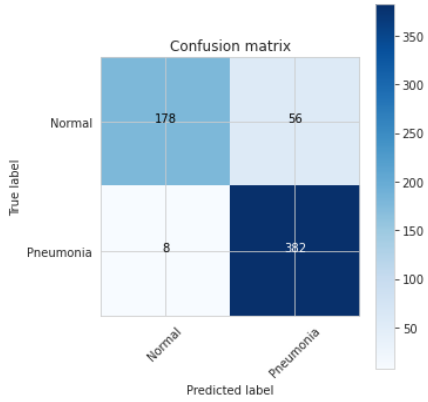


Figure 13 Vgg-19 validation set

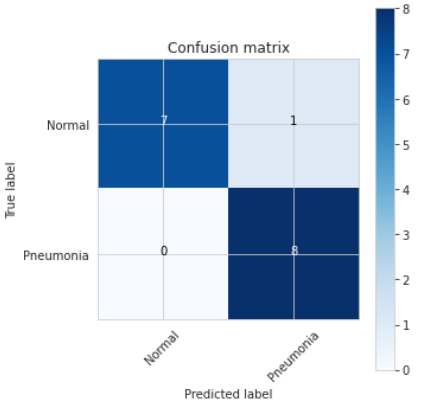


Figure 14 Vgg-19 Test set

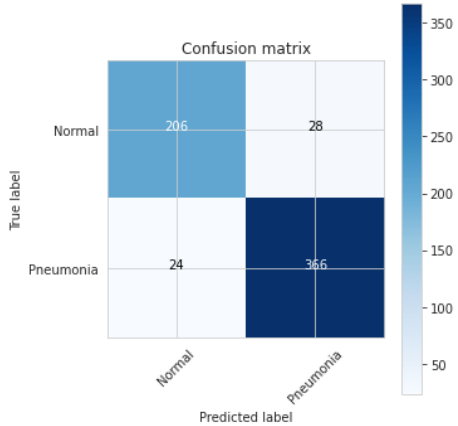


Figure 15 Vgg-19 -After Dropout - Validation set

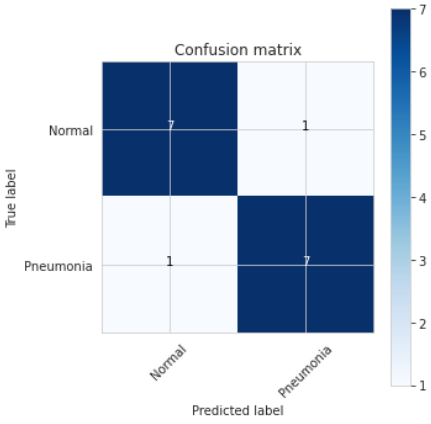


Figure 16 Vgg-19 - After dropout - Test set

Test set evaluation

In test set evaluation, we found that our precision, recall and f1 scores all are 0.88 as it failed to predict one true positive class and one true negative class.

NASNetMobile

NASNet, by Google Brain, is reviewed. Authors propose to *search for an architectural building block on a small dataset* and then *transfer the block to a larger dataset*. Particularly, they search for the best convolutional layer or cell on CIFAR-10 first, then apply this cell to the ImageNet by stacking together more copies of this cell. A new regularization technique called *ScheduledDropPath* is also proposed which significantly improves the generalization in the NASNet models. At last, NASNet model achieves state-of-the-art results with smaller model size and lower complexity (FLOPs) [3].

Another model we used is NASNetMobile architecture. In our model we used this pretrained model. Then we flatten the the output of the pretrained model. We then used two dense layer. First one with 128 units and the other is 2 units for classification. The input shape was 224*224*3 image. We used softmax as activation function in the last layer. We used 10 epochs for training where each epoch had 50 steps. The batch size was 32. This model had 6,623,618 trainable parameters and 4,269,716 non-trainable parameters.

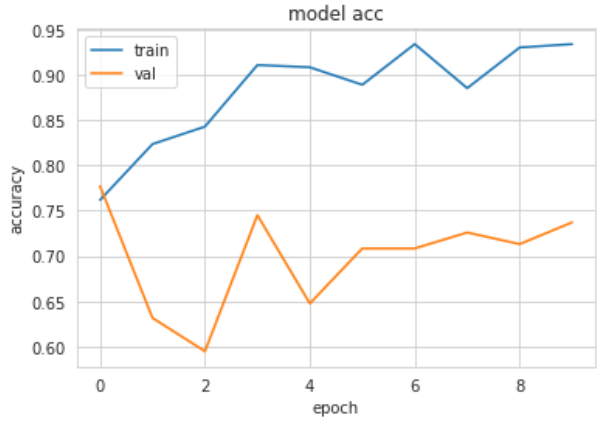


Figure 17 NASNetMobile

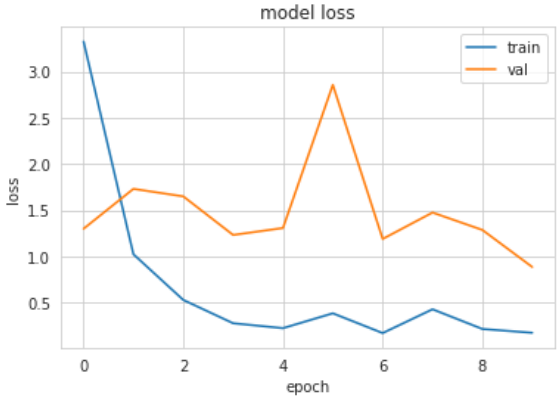


Figure 18 NASNetMobile

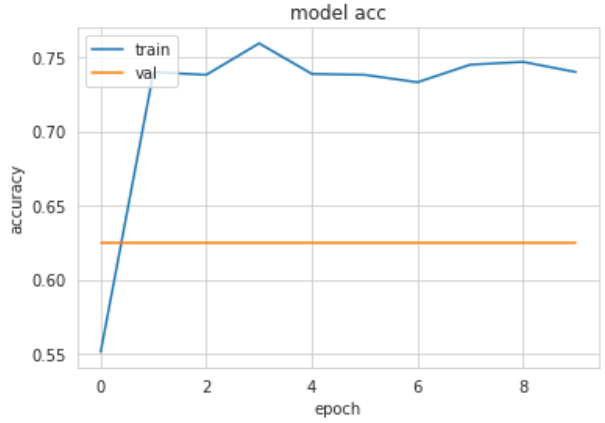


Figure 19 NASNetMobile After dropout

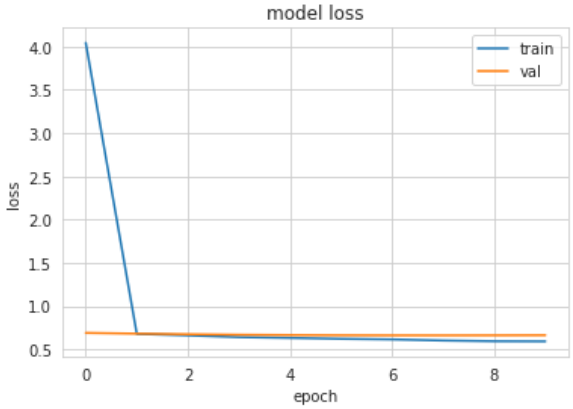


Figure 20 NASNetMobile After dropout

Validation set evaluation

Though the model achieved 0.92 accuracy in training set, the accuracy of validation set was 0.73. The recall score was 0.86 whereas the precision was 0.75. The model achieved 0.80 f1 score. In confusion matrix, we can see that the model did ok in predicting true positive but in 113 samples of Normal class, it predicted Pneumonia.

Test set Evaluation

In test set the result was not good. We can see the accuracy in test is 50 percent. The interesting fact here is recall value of this model is 1.00. The precision score 0.5 and f1 score is 0.67. The model did pretty well in predicting "Pneumonia class". In fact, it predicted correctly for all samples. However, the accuracy of predicting normal class was 0. It could not predict any single "Normal" class correctly.

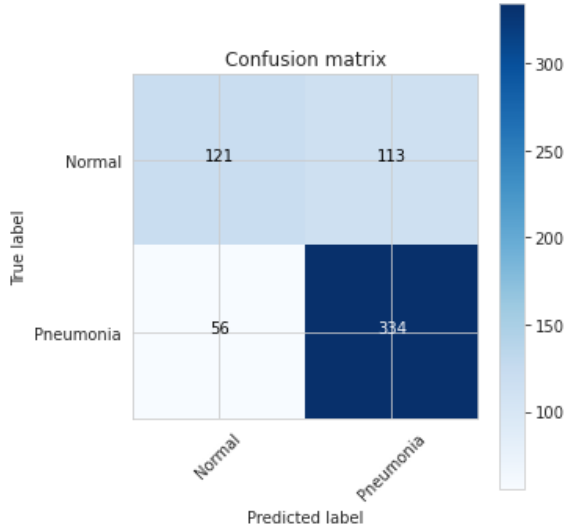


Figure 21 NASNetMobile Validation

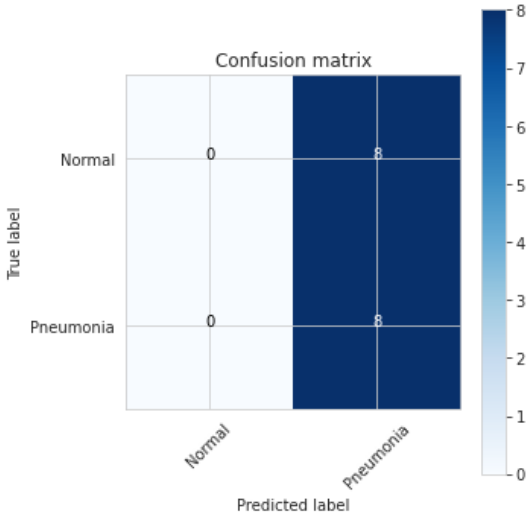


Figure 22 NASNetMobile Test set

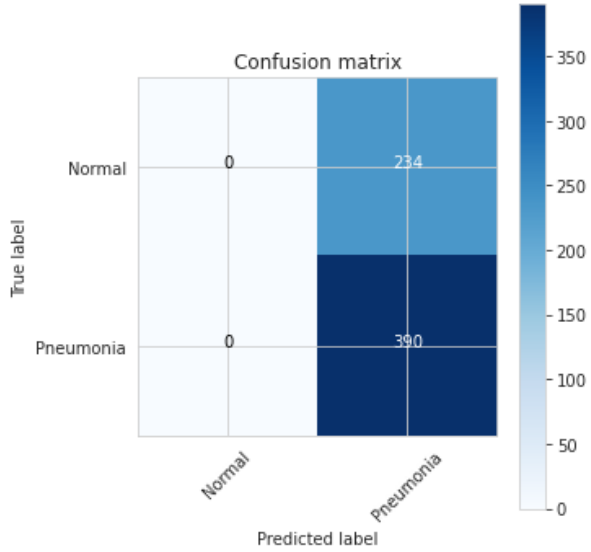


Figure 23 NASNetMobile After dropout Validation set

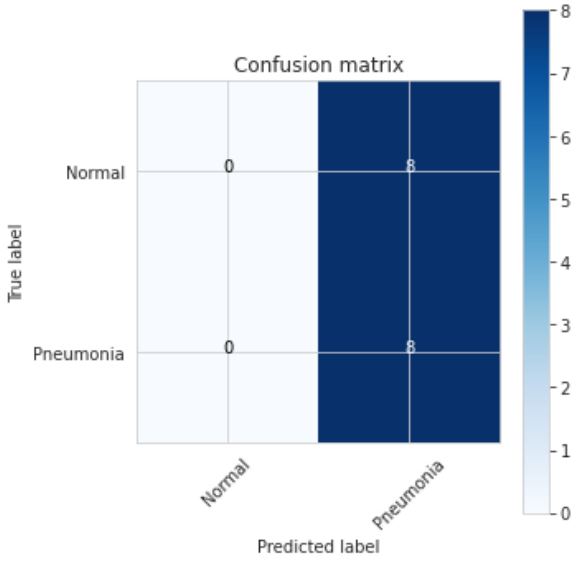


Figure 24 NASNetMobile After dropout Test set

NASNetMobile After Dropout

As the training accuracy was high and validation accuracy was less, we tried to experiment the result with dropout layer. We added one dropout layer with 0.4 rate after NasNetMobile pretrained model. Rest of the parameters were same.

Validation set evaluation

After running dropout, both training and testing accuracy decreased compared to NasNetMobile model without dropout. The validation accuracy decreased to 0.62. This model failed to predict the "Normal" class for any of 234 sample though it predicted correctly all "Pneumonia" class. So the recall became 1.00 but f1-core 0.77.

Test set evaluation

In test set, the accuracy was 50 percent as there were equal number of sample of both class in test set. Like, NasNetMobile model, the recall score was 1.0, precision was 0.50 and the f1-scoe was 0.67 in test set. This model resulted pretty bad predicting "Normal" class. The False Positive rate is very high.

MobileNet

MobileNet is a type of convolutional neural network designed for mobile and embedded vision applications. They are based on a streamlined architecture that uses depthwise separable convolutions to build lightweight deep neural networks that can have low latency for mobile and embedded devices [4].

The next model we used is MobileNet. After using the pretrained model we used one flatten layer and two dense layer with 128 units and 2 units respectively. The output layer had softmax activation function. The input image size was 224*224*3. This model had 6,422,914 Trainable params and 3,228,864 non-trainable parameters. In this model, 15 epcohs were used where each epoch had 100 steps for training.

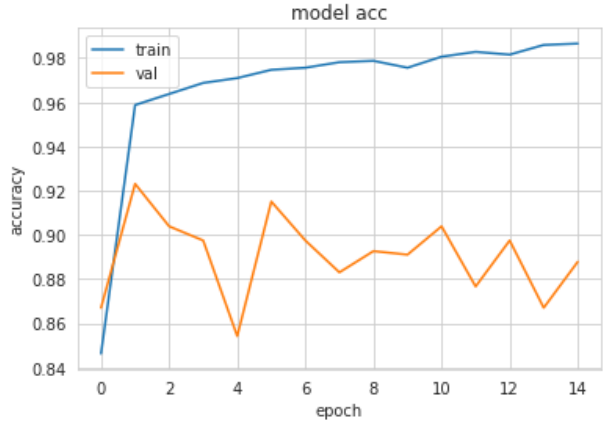


Figure 25 MobileNet

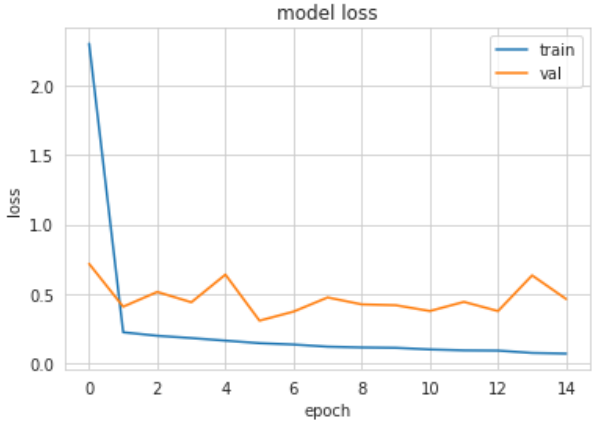


Figure 26 MobileNet

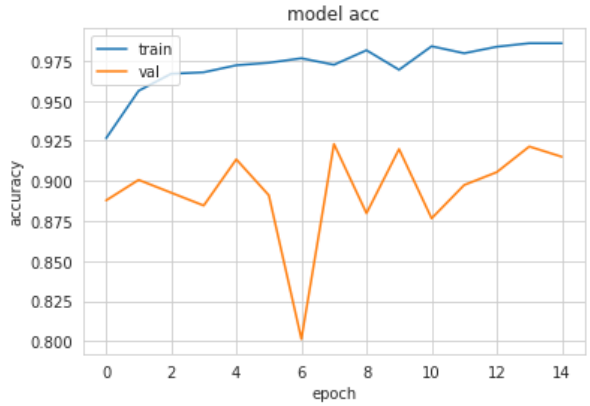


Figure 27 MobileNet After Dropout

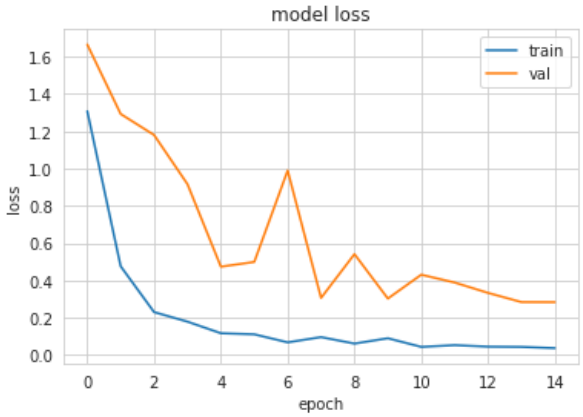


Figure 28 MobileNet After Dropout

Validation set Evaluation

In validation set we got 0.89 accuracy. The precision was 0.86 and recall was 0.99. The f1 score was 0.92. The model was good at predicting True positive class. Only 5 samples among 390 samples of "Pneumonia" were mis-classified.

Test set Evaluation

In Test set the accuracy is 0.94. The precision was 0.89 and recall was 1.00. The f1 score was 0.94. Here all the samples of "Pneumonia" were correctly classified and 1 class of "Normal" class was mis-classified.

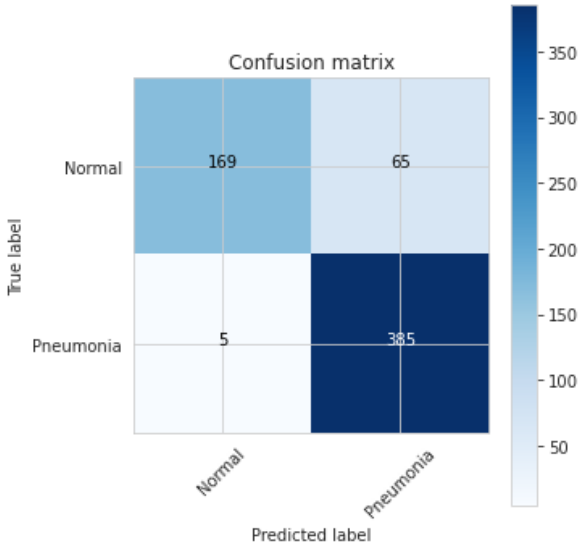


Figure 29 MobileNet Validation set

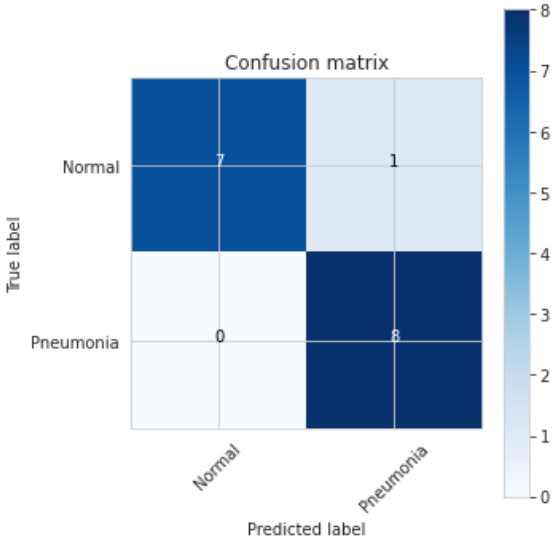


Figure 30 MobileNet Test set

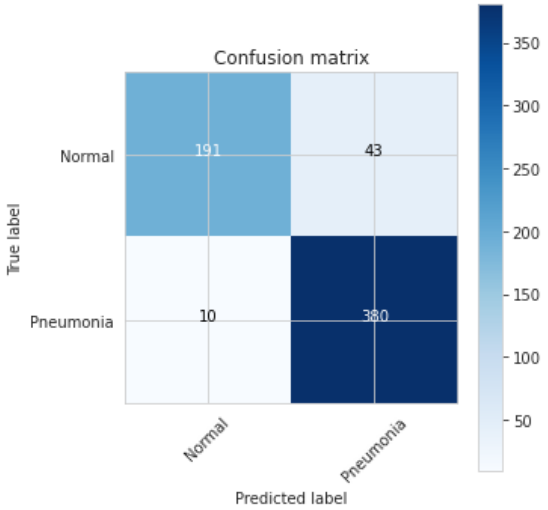


Figure 31 MobileNet - After dropout- Validation set

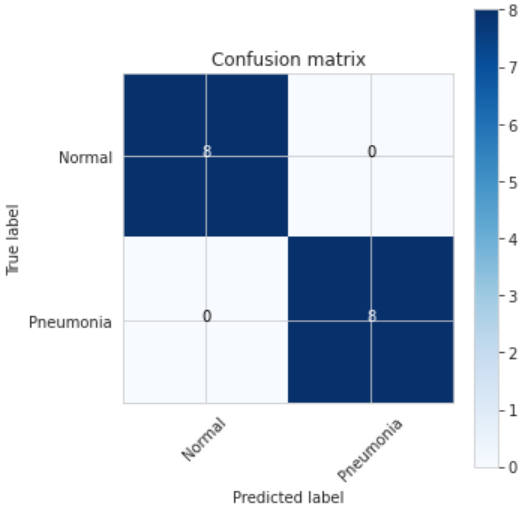


Figure 32 MobileNet - After dropout- Test set

MobileNet After Dropout

A dropout layer we introduced after pretrained layer with 0.4 rate. We used the same epoch number and steps per epoch as the MobileNet. The drop out rate made the training accuracy as 0.98 and validation accuracy to 0.92 which is 3% better than MobileNet model.

Validation Set evaluation

In validation set, the precision increased to 0.9. However, there was slight decrease in recall rate which is 0.97. The f1 score becomes 0.93. The misclassification of true positive class becomes 10 where as it was 5 without dropout.

Test set evaluation

In test set, we got the best result. The accuracy came to 1.0. The precision, recall and f1 score becomes 1.0 for MobileNet model with dropout.

MobileNetV2

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. The architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers [5].

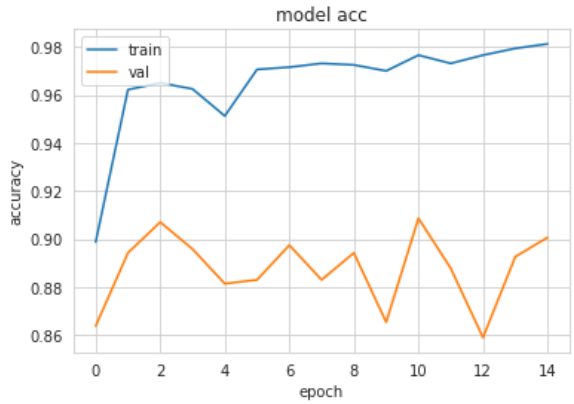


Figure 33 MobileNetV2

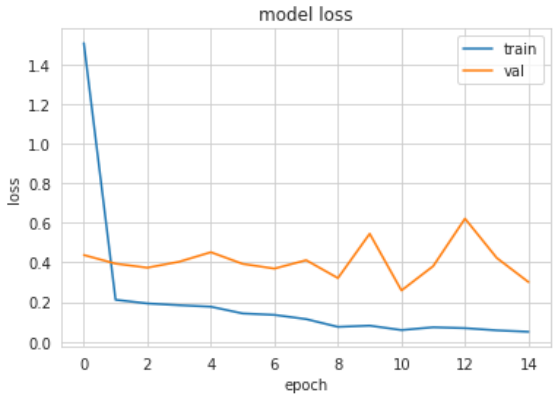


Figure 34 MobileNetV2

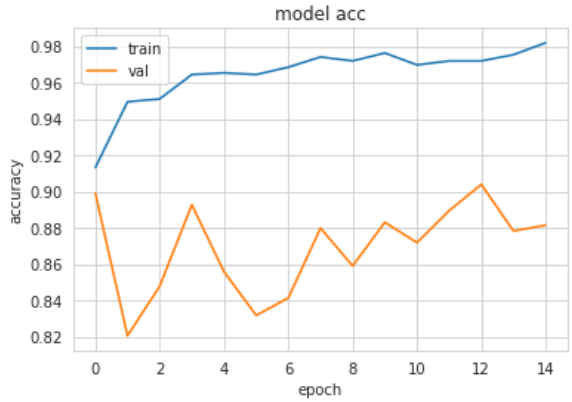


Figure 35 MobileNetV2 After dropout

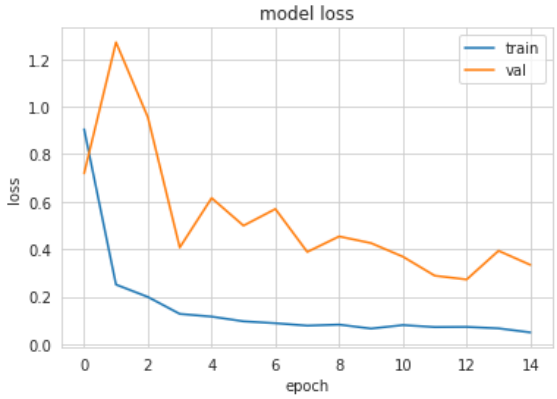


Figure 36 MobileNetV2 After Dropout

We used MobileNetV2 pretrained network with a flatten layer and two dense layers. We used softmax activation layer in the output layer. We used 15 epochs and 100 steps per epoch for training the model. The training accuracy was 0.98 and validation accuracy becomes 0.9.

Validation set Evaluation

In validation set we got 0.90 accuracy. The precision score was 0.9 and recall score was 0.94. The f1-score was 0.92.

Test set evaluation

In test set we also got 100 percent accuracy as MobileNet. The precision, recall, and f1-score is also 100 percent.

MobilenetV2 After Dropout

We used 0.4 rate of dropout after the pretrained model. That made the model training accuracy is 0.98 and validation accuracy is 0.88. In this model the dropout rate did not help to increase the validation accuracy.

Validation set evaluation

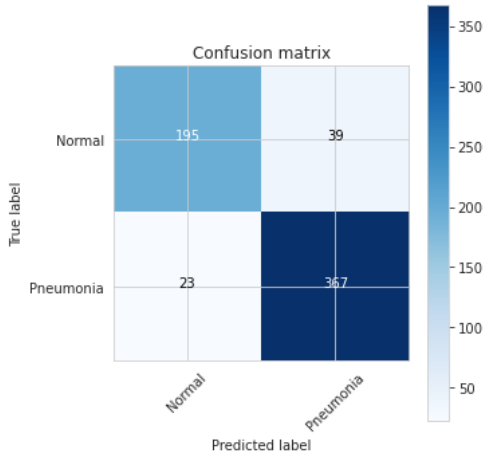


Figure 37 MobileNetV2 Validation set

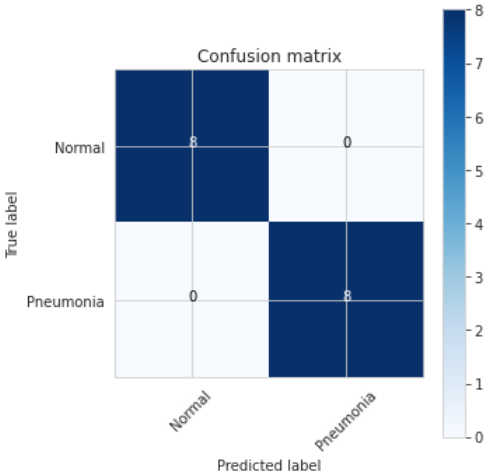


Figure 38 MobileNetV2 After dropout Test set

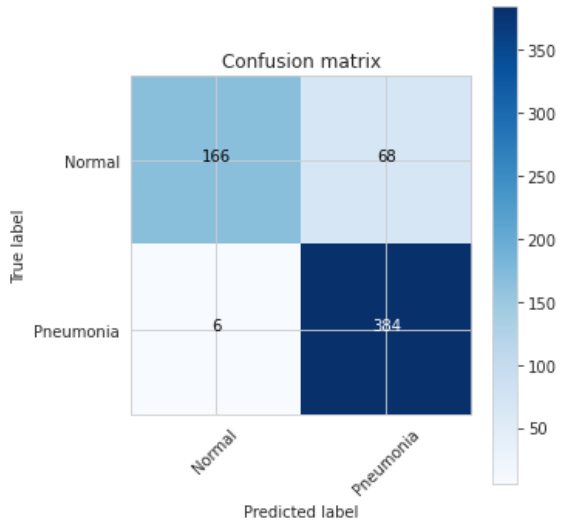


Figure 39 MobileNetV2 After dropout Validation set

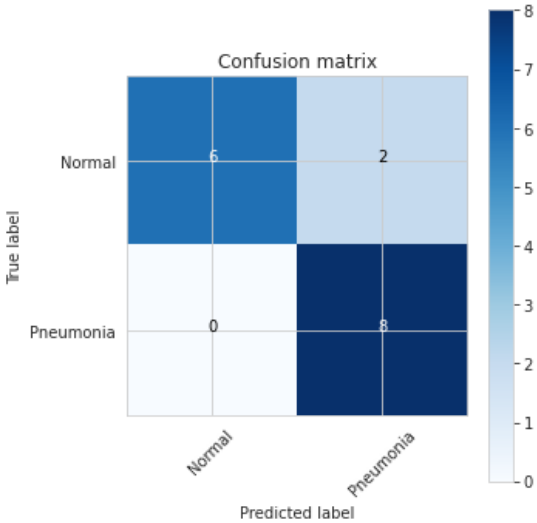


Figure 40 MobileNetV2 After dropout Test set

In validation set evaluation, we get precision 0.85 and recall is 0.98. Here the recall value increased by 4 percent. Thef1-score of this model was 0.91.

Test set Evaluation

In test set evaluation, we got 0.88 accuracy. The precision score is a bit low with 0.8 but recall is 1.0. Here the f1-score is 0.89.

Xception

Xception is a deep convolutional neural network architecture that involves Depthwise Separable Convolutions. It was developed by Google researchers. Google presented an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depthwise separable convolution operation [6].
The pretrained Xception model was used along with one flatten layer and two dense layer. The dense layers have 128 and 2 units respectively. This model was trained for 10 epochs with 50 steps per epochs. The model had 12,845442 trainable parameters and 20,861,480 non trainable parameters.

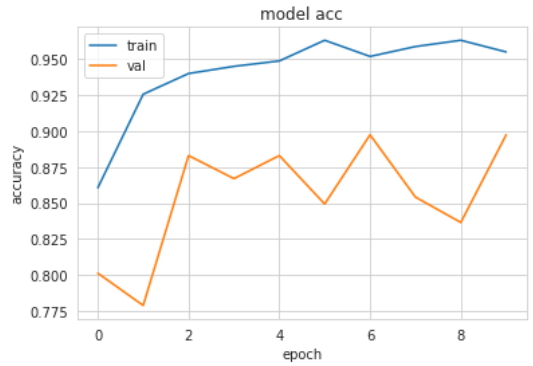


Figure 41 Xception

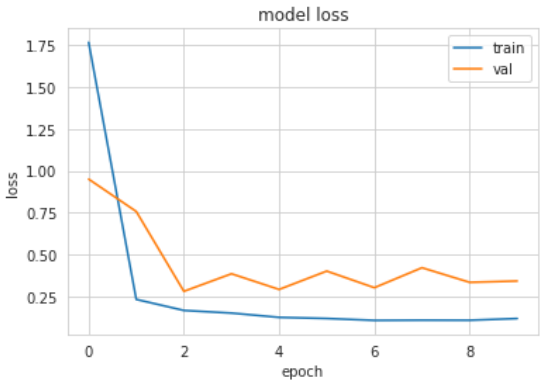


Figure 42 Xception

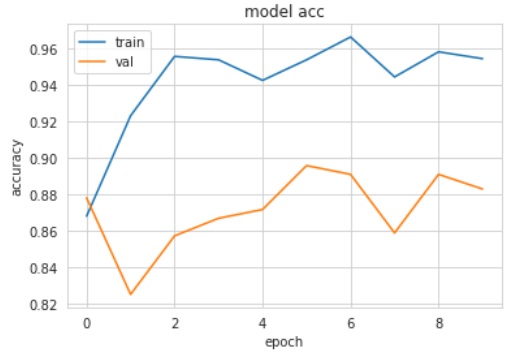


Figure 43 Xception After Dropout

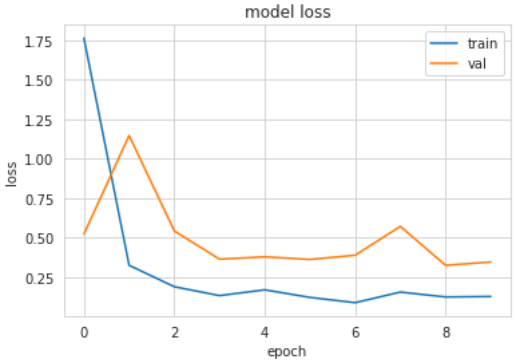


Figure 44 Xception After Dropout

Validation set evaluation

In validation set, The precision rate is 0.86 and recall rate is 0.97. We have got f1 score 0.91.

Test set evaluation

In test set, The accuracy rate is 0.88. The precision rate is 0.78 and recall is 0.88. We have got the f1-score 0.82.

Xception After Dropout

0.4 dropout rate was introduced after Xception pretrained layer. Other configuration was same as Xception model. The training accuracy went to 0.94 and validation accuracy to 0.89.

Validation set Evaluation

In validation set, The precision rate is 0.89 which 3 percent higher than without dropout and recall rate is 0.95. The recall rate decreased compare to the network without dropout. We have got f1 score which increased to 0.92.

Test set evaluation

In test set, The accuracy rate is 0.88. The precision rate is 0.88 and recall is 0.88. We have got the f1-score 0.88 which is better than the layer in terms of precision rate and f1 score.

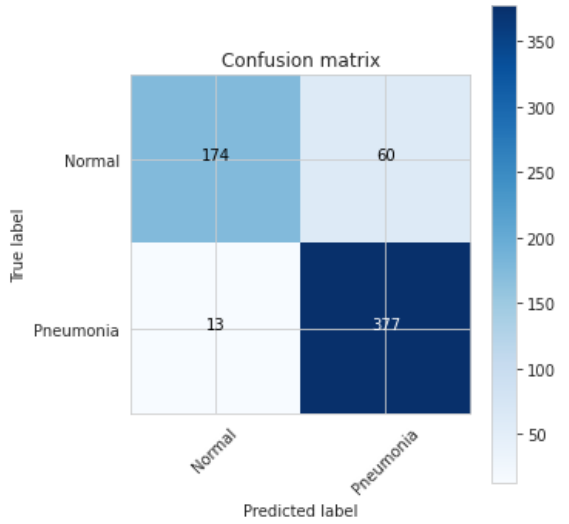


Figure 45 Xception Validation set

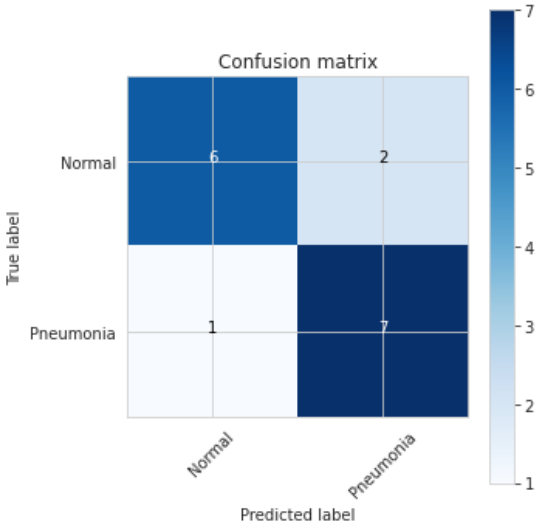


Figure 46 Xception Test set

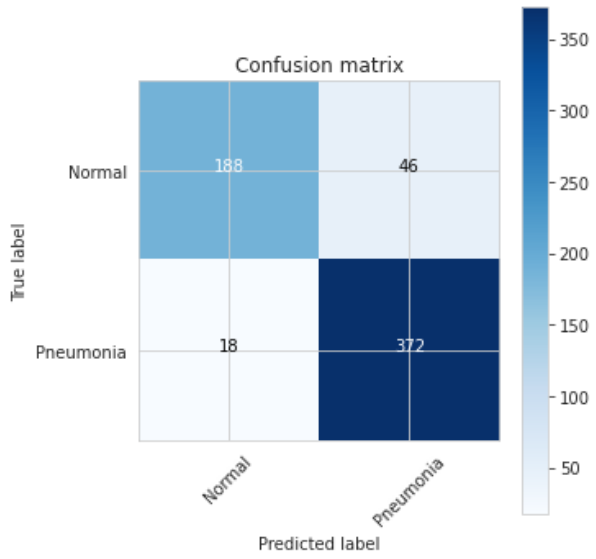


Figure 47 Xception After Dropout Validation set

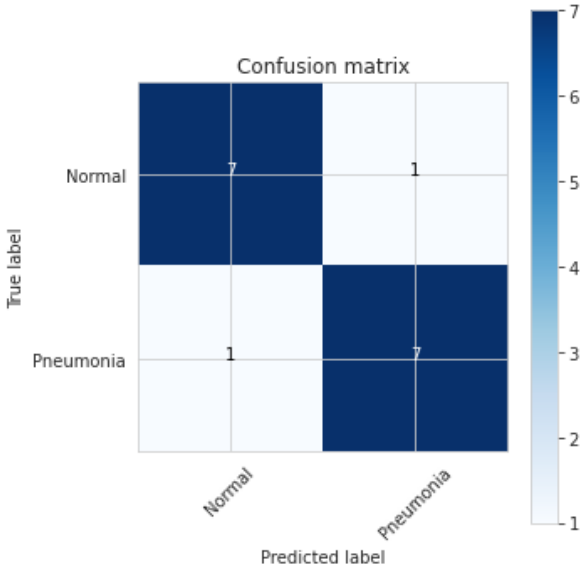


Figure 48 Xception After Dropout Test set

Custom made CNN model

CNN(7 Layers)

Apart from ecperimenting with pretrained model, we also run some experiments with custom CNN layers. The first experiment had 7 layers. The input shape here was 64*64*3. Two sets of layers were in the model where one Colvolution layer was followed by a max_pooling layer. Then one flatten layer was there and two dense layers. There were 14,626 trainable parameters. We used 15 epochs and 163 steps per epochs for training.

Callbacks

Keras call backs functionality was introduced here to dynamically change learning rate and save model checkpoint. Keras Earlystopping functionality is also used where the model would stop training if validation loss hold out for 10 epochs. The ReduceLROnPlateau function was there where validation accuracy was monitored. If the validation accuracy is not improving for 4 epochs the learning rate reduced by 0.2 factor where the minimum learning rate can decrease to 0.001.

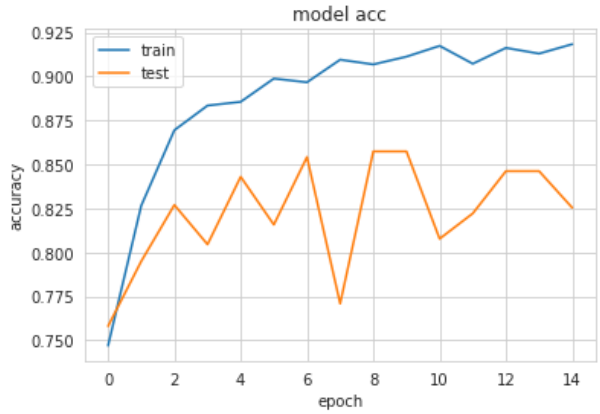


Figure 49 CNN-7 Layers

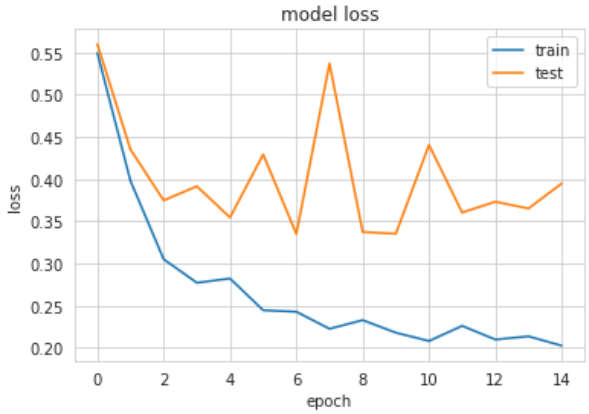


Figure 50 CNN-7 Layers

Validation set Evaluation

In validation set, we got 0.83 accuracy, the precision was 0.8 and recall 0.95. F1-Score was 0.87.

Test set Evaluation

In validation set, we got 0.83 accuracy, the precision was 0.8 and recall 0.95. F1-Score was 0.87.

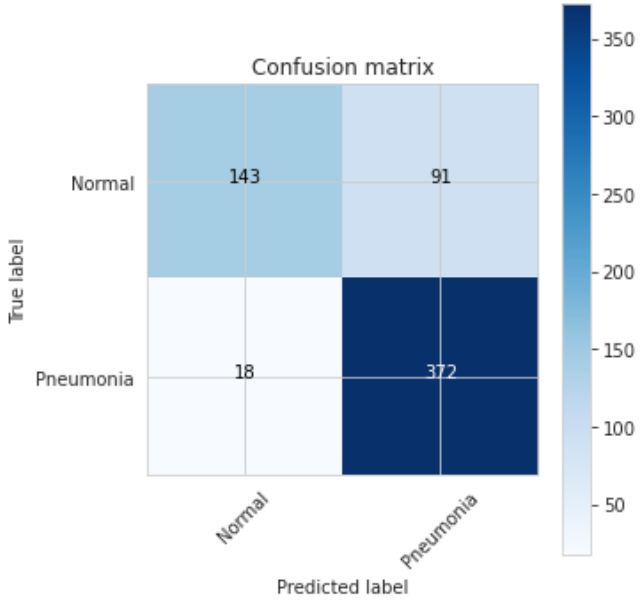


Figure 51 CNN - 7 Layers - Validation set

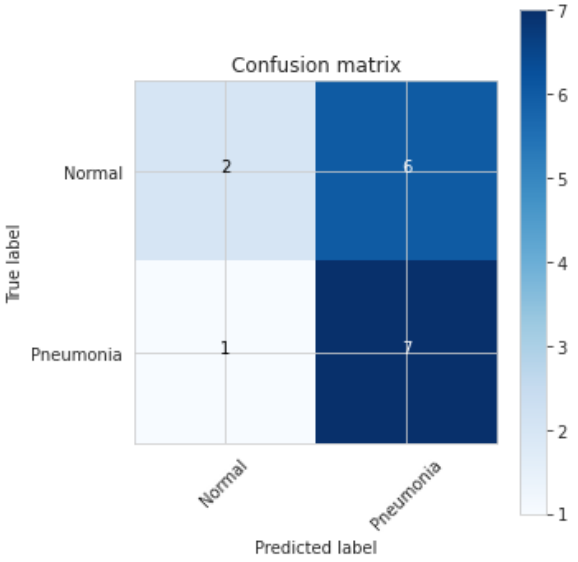


Figure 52 CNN - 7 Layers - Test set

CNN (17 layers)

In the next experimentation, we used 17 layers CNN to train our dataset. We have 4 block repetitive layers where one block contained a convolution layer, batch normalization or dropout and a maxpooling layers. After the 4 blocks a flatten layers is used. Two dense layers was used after the flatten layer. All these layers comprised 557,890 training parameters and 1,088 non trainable parameters. We used 15 epochs and 163 steps per epochs for training. We used the exact same callback used in previous CNN network.

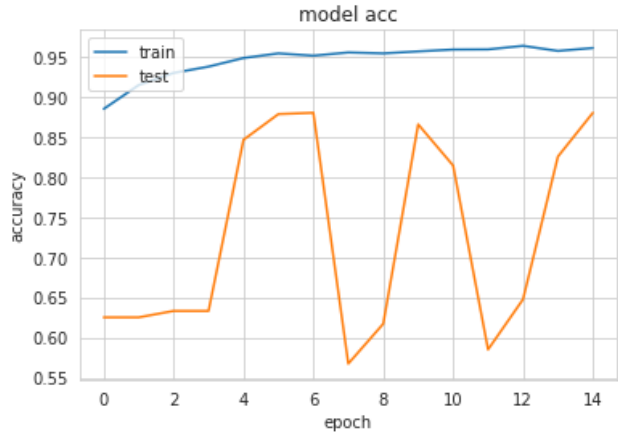


Figure 53 CNN - 17 layers

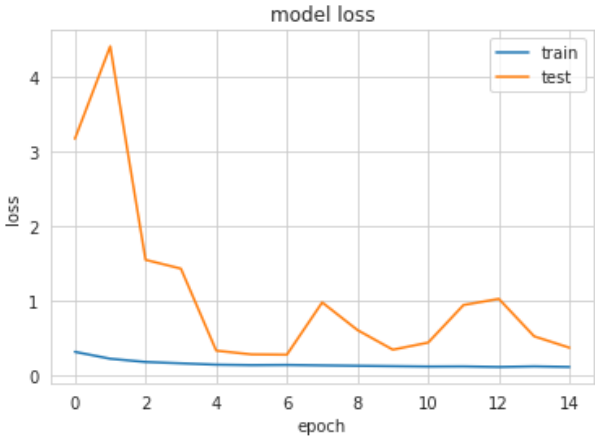


Figure 54 CNN-17 Layers

Validation set Evaluation

In validation set, we got 0.88 accuracy, the precision was 0.84 and recall 0.99. F1-Score was 0.91.

Test set Evaluation

In validation set, we got 0.75 accuracy, the precision was 0.75 and recall 0.75. F1-Score was 0.75.

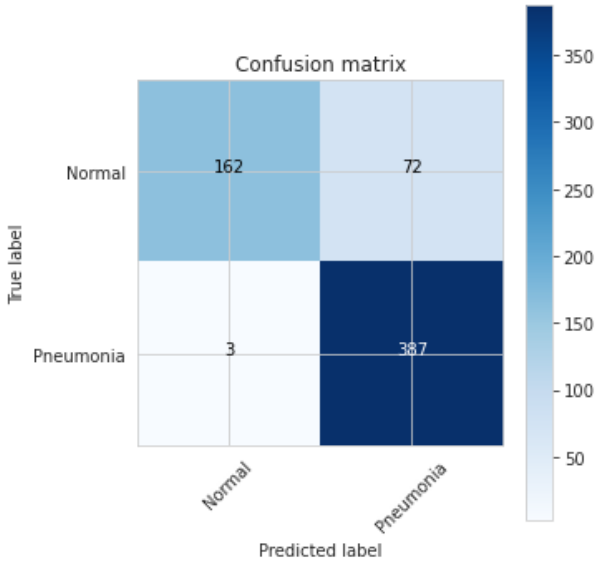


Figure 55 CNN-17 Layers - Validation set

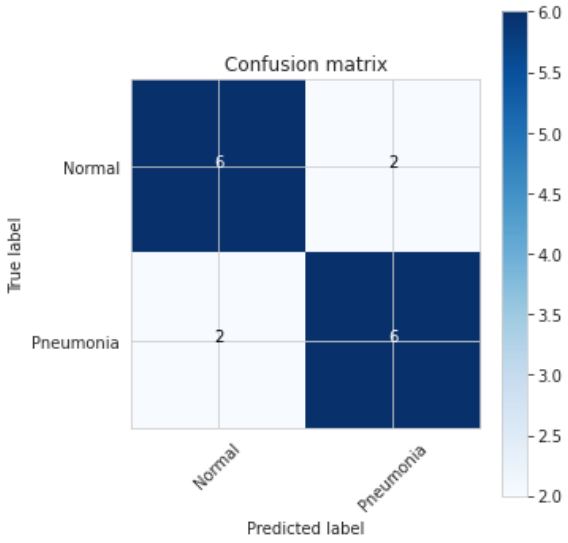
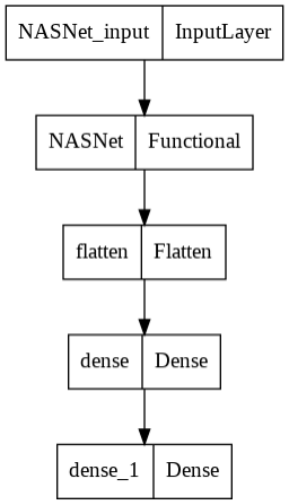


Figure 56 CNN-17 Layers - Test set

NASNetLarge

The convolutional neural network NASNet-Large was trained on over a million photos from the ImageNet collection. The network can sort photos into 1000 different object categories, including keyboards, mice, pencils, and a variety of animals. As a result, the network has learnt a variety of rich feature representations for a variety of pictures. The network's picture input size is 331 × 331 pixels.

We used three extra layers in this model after utilizing the NASNET-Large pre-trained model. We added two dense layers with 128 and 2 units each after flattening the output of the pretrained model using a flatted layer. Our model now has **62,448,002** trainable parameters and **84,916,818** non-trainable parameters due to the addition of these three layers.



Layer (type)	Output Shape	Param #
NASNet (Functional)	(None, 11, 11, 4032)	84916818
flatten (Flatten)	(None, 487872)	0
dense (Dense)	(None, 128)	62447744
dense_1 (Dense)	(None, 2)	258

Total params: 147,364,820
Trainable params: 62,448,002
Non-trainable params: 84,916,818

For training we used 10 epochs and each epoch consists of 15 steps. We chose ‘Adam’ as optimizer for the model. We can see from the graph that loss has been decreased with the increasing number of epochs, though there has been

some ups and down throughout the loss function for **validation data** (yellow line). But there was smooth transition of decreasing loss for training data. (Blue line). Interestingly, the accuracy function did not change much with epochs increasing through time for both training and validation data.

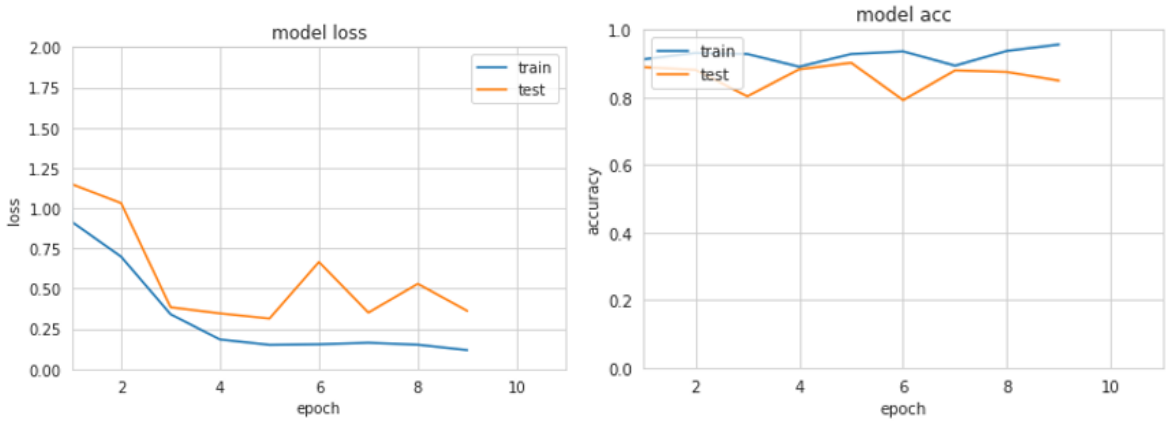


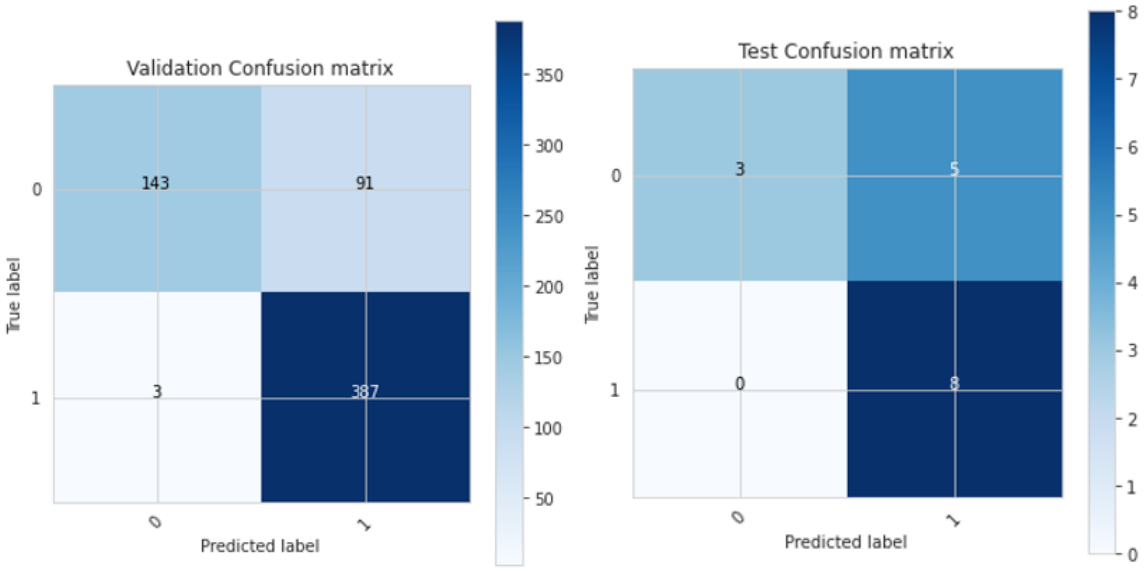
Figure: (Left) Loss function , (Right) Accuracy function

The least value of training loss 0.1179 we observed was at the end of the experiment. And we got maximum training accuracy **0.9563** also at the 10th epoch. But for validation data, we got the least loss value 0.3137 at the 6th epoch. Interestingly maximum accuracy 0.9022 for validation data was also at the 6th epoch. We could say the data were well trained and validation accuracy is also very good.

Validation set Evaluation

We are seeing from the following table that precision is 0.87 where f1 score (0.84) is also reasonably good. But recall value (0.61) for "NORMAL" class is quite less than that of "PNEUMONIA" class.

	precision	recall	f1-score	support
0	0.98	0.61	0.75	234
1	0.81	0.99	0.89	390
accuracy			0.85	624
macro avg	0.89	0.80	0.82	624
weighted avg	0.87	0.85	0.84	624



Test set Evaluation

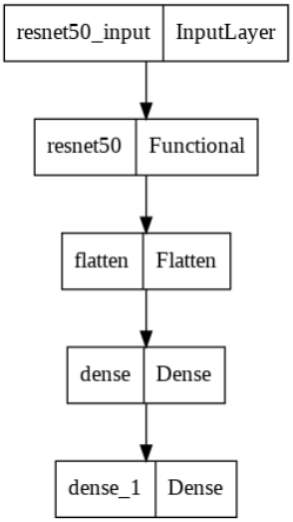
Here we can see that f1-score (0.65) did not provide value as good as for the validation set.

	precision	recall	f1-score	support
0	1.00	0.38	0.55	8
1	0.62	1.00	0.76	8
accuracy			0.69	16
macro avg	0.81	0.69	0.65	16
weighted avg	0.81	0.69	0.65	16

ResNet50

ResNet-50 is a 50-layer deep convolutional neural network. A pretrained version of the network trained on over a million photos from the ImageNet database may be loaded [1]. The pretrained network can categorize photos into 1000 different object categories, including keyboards, mice, pencils, and a variety of animals. As a consequence, the network has learnt detailed feature representations for a diverse set of pictures. The network's picture input size is 224 by 224.

After using the ResNet50 pre-trained model, we added three more layers to this model. After flattening the output of the pretrained model with a flatted layer, we added two dense layers of 128 and 2 units, respectively. Because of the inclusion of these three layers, our model now includes 12,845,442 trainable parameters and 23,587,712 non-trainable parameters.



Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 128)	12845184
dense_1 (Dense)	(None, 2)	258

Total params: 36,433,154
Trainable params: 12,845,442
Non-trainable params: 23,587,712

For training we used 10 epochs and each epoch consists of 30 steps. We chose 'Adam' as optimizer for the model. We can see from the graph that loss has been decreased with the increasing number of epochs, and though there has not been any sharp change in the loss function. Loss function for **validation data** (yellow line) also follows the that of training data. Interestingly, the accuracy function did not change much with epochs increasing through time for both training and validation data. But accuracy of validation data is lower than the training data.

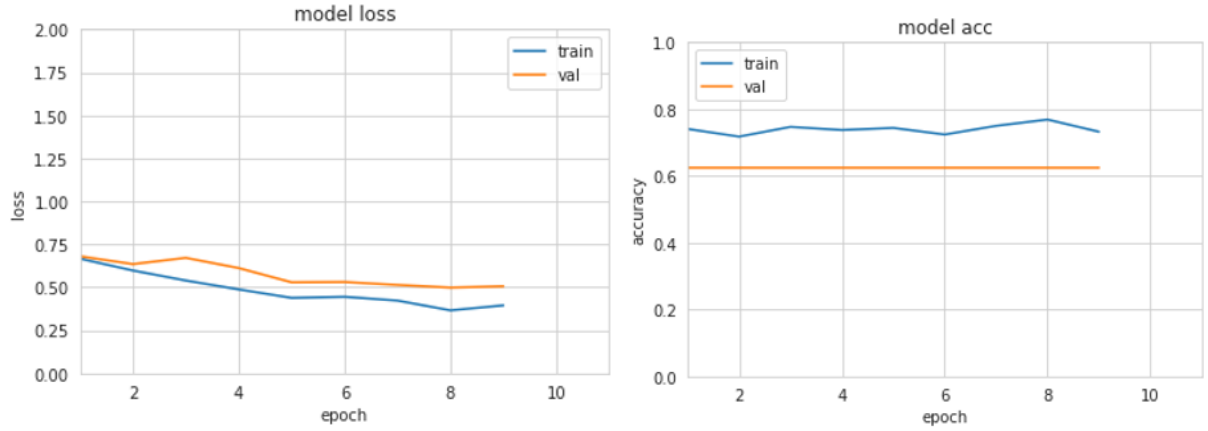


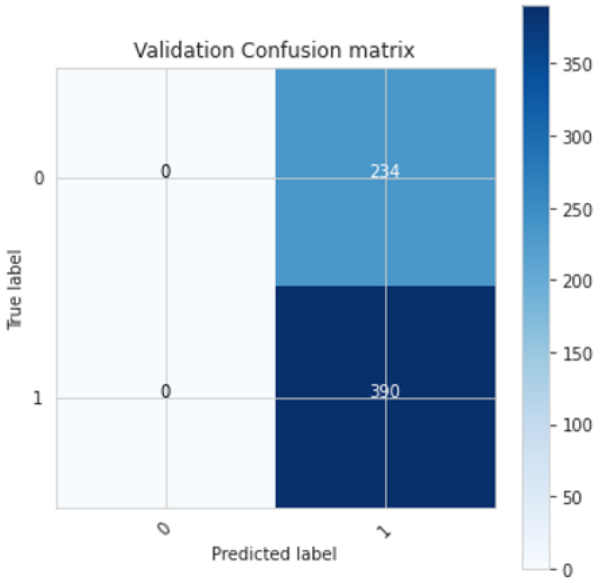
Figure : (Left) Loss function , (Right) Accuracy function

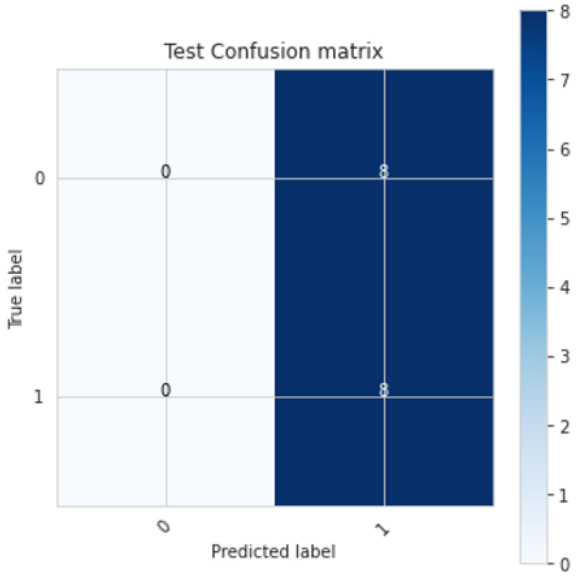
The least value of training loss 0.3658 we observed was at the 9th epoch of the experiment. And we got maximum training accuracy 0.7688 also at the 9th epoch. For validation data, we got the least loss value 0.4982 at the 9thepoch again. Interestingly there was not any change in accuracy with increasing epoch for validation data. Here actually this model was not effective enough to get the better result. Validation result did not get better with the increase in epoch number.

Validation set Evaluation:

From the following table, we can get that this model did not give good f1-score. Interestingly this model completely misclassified the 'NORMAL' data.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	234
1	0.62	1.00	0.77	390
accuracy			0.62	624
macro avg	0.31	0.50	0.38	624
weighted avg	0.39	0.62	0.48	624





Test Set Evaluation

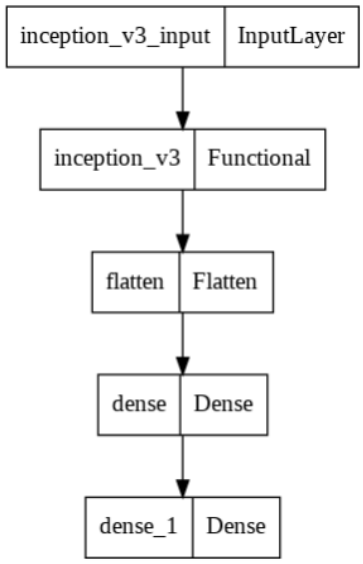
Test set also followed almost the same accuracy, precision and f1 score as that of validation set.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.50	1.00	0.67	8
accuracy			0.50	16
macro avg	0.25	0.50	0.33	16
weighted avg	0.25	0.50	0.33	16

Inception-v3

Inception-v3 is a convolutional neural network that is 48 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The input size of this model was 224 by 224.

We added three extra layers to this model after utilizing the Inception-v3 pre-trained model. We added two dense layers of 128 and 2 units, respectively, after flattening the output of the pretrained model with a flatted layer. Our model now has 6,553,986 trainable parameters and 21,802,784 non-trainable parameters due to the addition of these three layers.



Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
flatten (Flatten)	(None, 51200)	0
dense (Dense)	(None, 128)	6553728
dense_1 (Dense)	(None, 2)	258

Total params: 28,356,770
Trainable params: 6,553,986
Non-trainable params: 21,802,784

We employed a total of ten epochs, each with five steps. For the model, we choose 'Adam' as the optimizer. The graph shows that within the 2nd period, the loss has dropped rapidly. Following that, there were some ups and downs for validation data, though no such ups and downs for training data. The accuracy function was also inversely proportional to the loss function. However, as the number of epoch increased, the results improved for both the loss and accuracy functions.

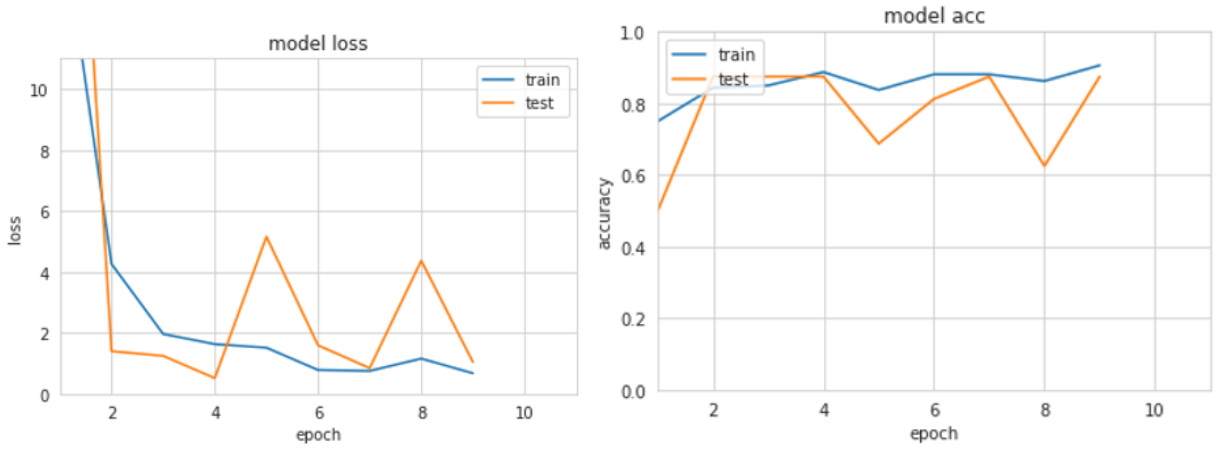


Figure: (Left) Loss function , (Right) Accuracy function

The least value of training loss 0.6734 we observed was at the end of the experiment. And we got maximum training accuracy 0.9062 also at the last epoch. For validation data, we got the least loss value 0.8414 at the 8thepoch. We got the highest accuracy at 3rd,4th,5th,8th and 10th epoch for validation data. This model has been trained really fast within fewer number of epochs. Loss and accuracy graph of both set of data (training and validation) show the promising result that can be effective for testing sets.

Validation set Evaluation

We can see from the following table that both class (NORMLA and PNEUMONIA) show very good result (0.88) in classifying accurately. Precision is 90% and f1-score is 87%. All the metrics for evaluation is very high.

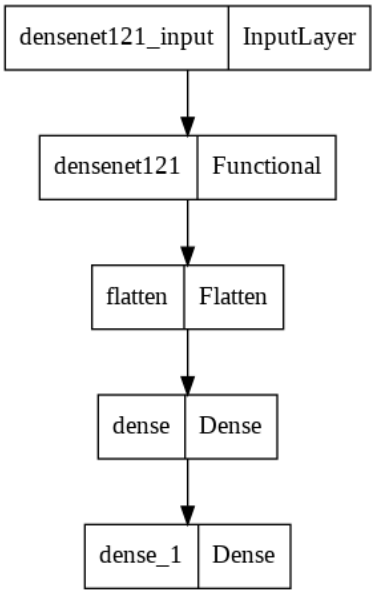
	precision	recall	f1-score	support
0	1.00	0.75	0.86	8
1	0.80	1.00	0.89	8
accuracy			0.88	16
macro avg	0.90	0.88	0.87	16
weighted avg	0.90	0.88	0.87	16

Test Set Evaluation

Test set also shows the same level of accuracy, precision, recall and f1-score as of validation set.

	precision	recall	f1-score	support
0	0.87	0.78	0.82	234
1	0.87	0.93	0.90	390
accuracy			0.87	624
macro avg	0.87	0.85	0.86	624
weighted avg	0.87	0.87	0.87	624

DenseNet121




```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 7, 7, 1024)        7037504

 flatten (Flatten)           (None, 50176)             0

 dense (Dense)               (None, 128)               6422656

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 13,460,418
Trainable params: 6,422,914
Non-trainable params: 7,037,504
_________________________________________________________________
```

We can see from the above figures that densenet121 has three more layers following the functional layer of dense121. Here number of trainable parameter is 6,422,914 whereas nontrainable parameter is 7,037,504.

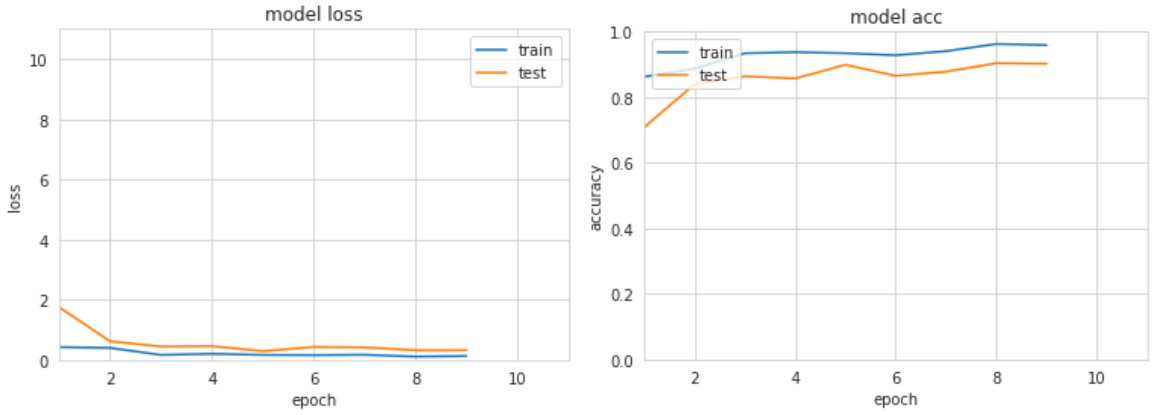
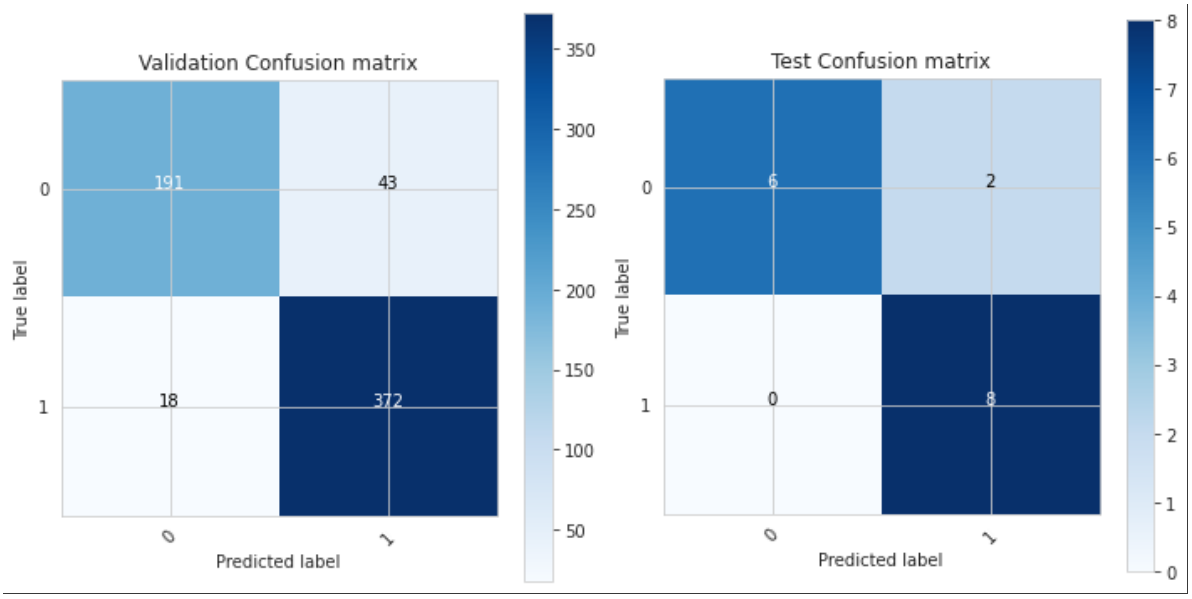


Figure : (Left) Loss function , (Right) Accuracy function

Total 10 epochs are applied where each epoch contains 10 step. We can see from the figure that running epoch multiple times did not make any better for the pretrained model. From the very beginning of the epoch the loss function value was pretty low for both training and validation data sets. And accuracy function was also near to 1 from the very first epoch. As a result we can say, this pretrained model with non trainable parameters are already good enough to provide good accuracy.

Validation set evaluation

	precision	recall	f1-score	support
0	0.91	0.82	0.86	234
1	0.90	0.95	0.92	390
accuracy			0.90	624
macro avg	0.91	0.89	0.89	624
weighted avg	0.90	0.90	0.90	624



Test Set Evaluation:

	precision	recall	f1-score	support
0	1.00	0.75	0.86	8
1	0.80	1.00	0.89	8
accuracy			0.88	16
macro avg	0.90	0.88	0.87	16
weighted avg	0.90	0.88	0.87	16

Comparative Analysis

Here is a table showing the four metrics (accuracy, precision, recall and f1 score) for each model applied on both validation and test data. This table also includes the Epochs and Steps per Epoch for each model used in training.

	Model Name	Epochs	Steps per epoch	Validation				test			
				accuracy	precision	recall	F1 score	accuracy	precision	recall	F1 score
1	VGG16	15	100	0.87	0.83	0.99	0.9	0.69	0.62	1	0.76
2	VGG16 Dropout	15	100	0.92	0.9	0.98	0.94	0.81	0.73	1	0.84
3	VGG19	15	100	0.9	0.87	0.98	0.92	0.94	0.89	1	0.94
4	VGG19 Dropout	15	100	0.92	0.93	0.94	0.93	0.88	0.88	0.88	0.88
5	Xception	10	50	0.88	0.86	0.97	0.91	0.81	0.78	0.88	0.82
6	Xception Dropout	10	50	0.9	0.89	0.95	0.92	0.88	0.88	0.88	0.88
7	MobileNet	15	100	0.89	0.86	0.99	0.92	0.94	0.89	1	0.94
8	MobileNet Dropout	15	100	0.92	0.9	0.97	0.93	1	1	1	1
9	MobileNetV2	15	100	0.9	0.9	0.94	0.92	1	1	1	1
10	MobileNetV2 Dropout	15	100	0.88	0.85	0.98	0.91	0.88	0.8	1	0.89
11	NasNetMobile	10	50	0.73	0.75	0.86	0.8	0.5	0.5	1	0.67
12	NasNetMobile Dropout	10	50	0.62	0.62	1	0.77	0.5	0.5	1	0.67
13	CNN (7 layers)	15	163	0.83	0.8	0.95	0.87	0.56	0.54	0.88	0.67
14	CNN (17 layers)	15	163	0.88	0.84	0.99	0.91	0.75	0.75	0.75	0.75
15	ResNet50	10	30	0.62	0.39	0.62	0.48	0.5	0.25	0.5	0.33

16	InceptionV3	10	5	0.88	0.9	0.88	0.87	0.87	0.87	0.87	0.87
17	DenseNet121	10	10	0.9	0.9	0.9	0.9	0.89	0.9	0.88	0.87
18	NASNetLarge	10	20	0.85	0.87	0.85	0.84	0.69	0.81	0.69	0.65

Table: Four metrics of evaluation for each model and both datasets (validation and test)

Analysis for validation dataset

For validation data set, the best F-1 score **(0.94)** we have found from the **VGG16 Dropout**. Originally this model without the dropout layer did not provide good result for two other metrics accuracy and precision. But after applying dropout layers, precision and accuracy value got increased. It also happened for VGG19 Dropout. All the best accuracy (0.92) come out from adding dropout layer. Overall, the most two decent models with good value of accuracy, precision, recall and f1-score are VGG16 Dropout and VGG19 Dropout.

Analysis for test dataset

Interestingly, for test datasets, the most two decent models have been overtaken by MobileNet Dropout and MobileNetV2. These two models have provided the best performance in each metrics. Surprisingly, those models (VGG16 Dropout,VGG19 Dropout) which provided best result on validation data set did not do justice on test datasets. But in comparison with all other models we have applied, the ones which we can consider in practice are :

1. MobileNet Dropout
2. MobileNetV2
3. VGG19
4. MobileNetV2 Dropout
5. DenseNet121

Conclusion

The purpose of this project was to find the suitable CNN models for image classification on X-ray datasets. We wanted to apply pretrained models adding some extra layers and also used fully connected general CNN of different layers. In this report, we have presented experiment architecture, training result, validation and test set evaluation for each model. Later we have showed a comparative analysis at the end. We have figured out that adding dropout layer has increased the accuracy to some pretrained models. But in testing, we have found that validation result is not always consistent with the testing result, though the inconsistent is not huge. At the end, we provided the top five models we have found in our whole experiment.

Reference

[1] Kermany D., Goldbaum M., Cai W. Large dataset of labeled optical coherence tomography (OCT) and chest X-Ray images 2018, 172, 1122–1131. *Cell.* 2018;172:1122–1131. doi: 10.1016/j.cell.2018.02.010. [PubMed] [CrossRef] [Google Scholar]

[2] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[3] Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8697-8710).

[4] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[5] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[6] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).