**Capstone Project - Milestone Report**

Classifying Chest X-Ray images of Normal vs Pneumonia patients

**Background and Problem**

Pneumonia is an infection of the lungs that may be caused by bacteria, viruses or fungi. The infection causes the lung’s air sacs (alveoli) to become inflamed and fill up with fluid or pus. That can make it hard for the oxygen you breathe in to get into your bloodstream. The people most at risk are infants and young children, adults 65 or older and people who have other health problems.

According to the World health Organization (WHO), pneumonia kills about 2 million children under 5 years of age every year and is consistently estimated as the single leading cause of childhood mortality (Rudan et al., 2008), killing more children than HIV/AIDS, malaria and measles combined (Adegbola, 2012). The WHO reports that nearly all cases (95%) of new onset childhood clinical pneumonia occur in developing countries, particularly in Southeast Asia and Africa. Bacterial and viral pathogens are the two leading causes of pneumonia (Mcluckie, 2009) but require very different forms of management. Bacterial pneumonia requires urgent referral for immediate antibiotic treatment, while viral pneumonia is treated with supportive care. Therefore, accurate and timely diagnosis is imperative. One key element of diagnosis is radiographic data, since chest X-rays are routinely obtained as standard of care and can help differentiate between different types of pneumonia. However, rapid radiologic interpretation of images is not always available, particularly in the low-resource settings where childhood pneumonia has the highest incidence and highest rate of mortality.

Hence, a computational approach to classify the X-rays is very crucial for timely care of the patients.

**Data set**

Data Source: <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>

<https://data.mendeley.com/datasets/rscbjbr9sj/3>



The normal chest X-ray (left panel) depicts clear lungs without any areas of abnormal opacification in the image. Bacterial pneumonia (middle) typically exhibits a focal lobar consolidation, in this case in the right upper lobe (white arrows). Lung consolidation occurs when the air that usually fills the small airways in your lungs is replaced with something else and is called lobar when it occurs on tone of the lobes of the lung. It infers alveolar spread of disease. Whereas viral pneumonia (right) manifests with a more diffuse ‘‘interstitial’’ pattern in both lungs - the connective tissue (interstitium) that forms the support structure of the alveoli (air sacs) of the lungs

The dataset contains X-Rays from children and is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

**Solution and Client**

This is a binary classification problem – classifying the X-rays as normal or pneumonia and help getting urgent care for the patients. This will help healthcare professionals to speed up diagnosis and get treatment started for patients with pneumonia. Detection and treatment of such abnormalities at an early stage can help save lives.

**Relevant Papers**

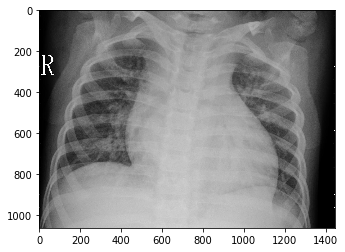
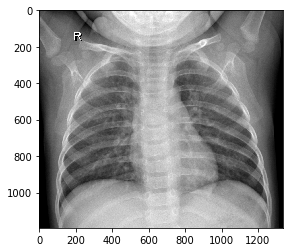
<http://www.cell.com/cell/fulltext/S0092-8674(18)30154-5>

Chhikara P., Singh P., Gupta P., Bhatia T. (2020) Deep Convolutional Neural Network with Transfer Learning for Detecting Pneumonia on Chest X-Rays. In: Jain L., Virvou M., Piuri V., Balas V. (eds) Advances in Bioinformatics, Multimedia, and Electronics Circuits and Signals. Advances in Intelligent Systems and Computing, vol 1064. Springer, Singapore

**Exploratory Data Analysis**

The data is stored in the ray folder in 3 sub folders – train, val and test. Each of them has a subfolder, one for NORMAL and one for PNEUMONIA. The data is imported into the hard drive using ‘glob’ and the image of a Normal and Pneumonia patient is loaded using the load\_img function from the Keras preprocessing package.

NORMAL PNEUMONIA



All the images are in the RGB mode and their pixel sizes are all varied. There are only 1341 Normal images in the training set as compared to the 3875 Pneumonia images. This is an imbalanced data set. There are 8 images of each type in the validation set. The test data has 234 Normal and 390 Pneumonia images.

The primary problem with an imbalanced data is that the majority class, in this case Pneumonia, gets highly biased and the model tends to predict everything as a Pneumonia image. There are many popular approaches to address this issue – Weighted class, Over-sampling, Under-sampling, Synthetic Minority Over-sampling Technique or SMOTE. SMOTE is an over-sampling technique that generates synthetic samples from the minority class. I have used SMOTE to address class imbalance.

The images are first combined into a single set containing the Normal and Pneumonia images. The images are first resized into 128, 128 pixel size and then converted into an array. The array is represented as (n\_samples, width, height, channels). The last 3 numbers are multiplied to represent the number of features and the array is flattened in to (n\_samples, features). This array is then used to fit into SMOTE. SMOTE is offered by the imblearn package in Python. Now, the number of Normal images is also 3875. The flattened image is then again converted to (n\_samples, width, height, channels) dimensions.

SMOTED image



The validation and test sets were also resized and stored into single folders containing both classes. The Y labels for all the data sets were also derived and stored as an array. Now, the train, val and test data are all ready for the neural network.

**Data Augmentation and Data Generators**

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class. This is used to expand the training set to improve performance and the ability of the model to generalize. The function ImageDataGenerator augments the image by iterating through image as your CNN is getting ready to process that image. Some of the features are rescaling, shearing, zooming and flipping the image (horizontally or vertically). We have employed only rescaling and horizontal flipping in the SMOTEd training set and only rescaling for its validation and test sets. When the data set is used without the SMOTE, another set of training, validation and test generators are prepared with data augmentations – rescaling, shearing, zooming and horizontal flips. The validation set and test set are only rescaled. The generators have a few methods to load the images – I used the flow method. This method creates the training set, val set and test set which comprises of the train generator, test generators for val and test respectively, in batches of 32 images.

Now the training sets (both SMOTed and non-SMOTEd) are ready to be fit into the different models for training. We use the validation set for the validation step during training. We can evaluate model performance on the test set.

**Model Architecture**

The **first model** is a simple Sequential CNN model. The different layers are: the Convolutional 2D layer with ReLU (Rectified Linear Unit) activation, MaxPooling2D layer, 2 Fully Connected layers and the output layer with Sigmoid activation. The model is compiled with the Adam optimizer, loss function is binary crossentropy and the metrics used to evaluate is Accuracy, Precision, Recall and AUC. While using the SMOTEd data set, Accuracy is acceptable.

The first model summary is as follows:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 126, 126, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 63, 63, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 61, 61, 32) 18464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 30, 30, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 28800) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 128) 3686528

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 1) 129

=================================================================

Total params: 3,706,913

Trainable params: 3,706,913

Non-trainable params: 0

The **second model** had 2 Dropout layers added to the first model – one after the last MaxPooling and one before the output layer.. Dropout layers randomly set a fraction rate of input units to 0 at each update during training time, which helps over fitting. Here the fraction was set to 0.5.

The second model summary is as follows:

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_2 (Conv2D) (None, 126, 126, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 63, 63, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 61, 61, 32) 18464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 30, 30, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 30, 30, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 28800) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 128) 3686528

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 1) 129

=================================================================

Total params: 3,706,913

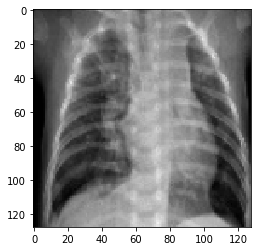
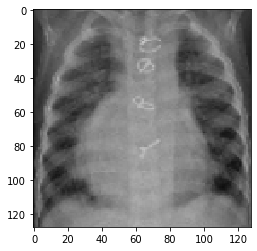
Trainable params: 3,706,913

Non-trainable params: 0

Both these models used the SMOTEd data set for training and the model was then used to evaluate the test set.

The **third model** architecture is the same as the second but the non-SMOTEd data was used to train. In this model, the non-SMOTEd data was used for more data augmentation features and the augmented image was also visualized to see if the images were comparable.

Examples of augmented image after rescaling, shearing, zooming and flipping:



The **fourth model** was developed where a 3rd Convolution layer and a 3rd Fully Connected layer were added to the second model. The summary is as follows:

Model: "sequential\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_6 (Conv2D) (None, 126, 126, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_6 (MaxPooling2 (None, 63, 63, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_7 (Conv2D) (None, 61, 61, 32) 18464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_7 (MaxPooling2 (None, 30, 30, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_8 (Conv2D) (None, 28, 28, 32) 9248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_8 (MaxPooling2 (None, 14, 14, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_4 (Dropout) (None, 14, 14, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_3 (Flatten) (None, 6272) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 128) 802944

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_5 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 64) 8256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_8 (Dense) (None, 1) 65

=================================================================

Total params: 840,769

Trainable params: 840,769

Non-trainable params: 0

**Tranfer Learning using InceptionV3 Model**

In addition to the models studied above, transfer learning was also employed. The following paper outlines the methods used with transfer learning and was also tried out.

<http://www.cell.com/cell/fulltext/S0092-8674(18)30154-5>

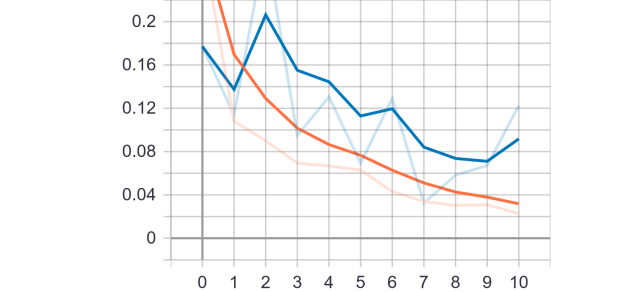
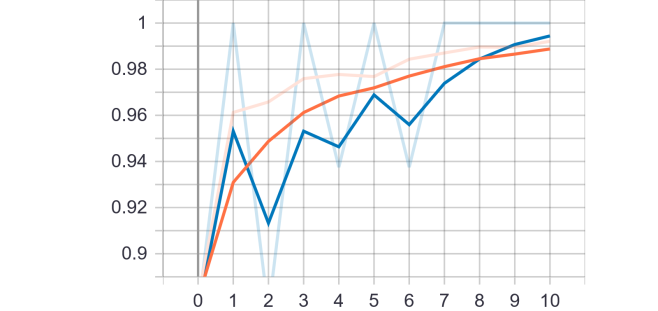
Chhikara P., Singh P., Gupta P., Bhatia T. (2020) Deep Convolutional Neural Network with Transfer Learning for Detecting Pneumonia on Chest X-Rays. In: Jain L., Virvou M., Piuri V., Balas V. (eds) Advances in Bioinformatics, Multimedia, and Electronics Circuits and Signals. Advances in Intelligent Systems and Computing, vol 1064. Springer, Singapore

Transfer learning was tried using the Inception V3 model and changing the output layer to match the current problem. We tried 4 different types of output layers with different combinations of activation and optimizer use – Sigmoid vs Softmax and Adam vs RMSprop. We got the models using softmax with RMS prop/Adam prop to give some result but the models with Sigmoid activation did not learn or converge.

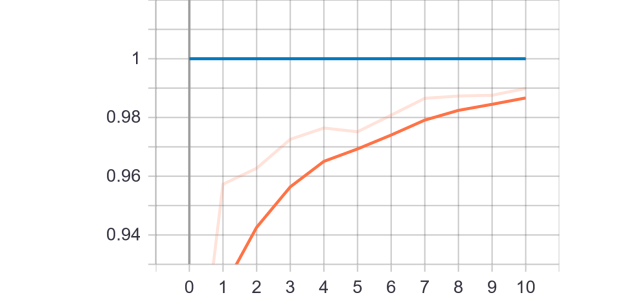
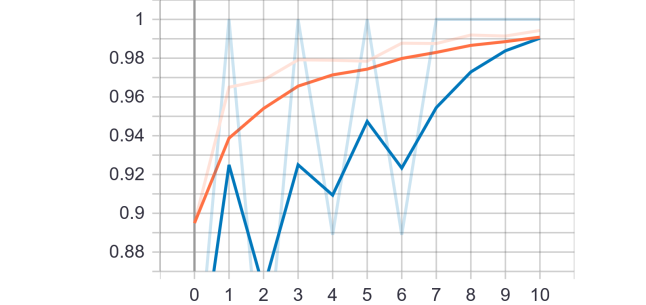
**Results**

**Model 1 – SMOTEd data with 2 convolutions, 2 fully connected layers**

Accuracy Loss

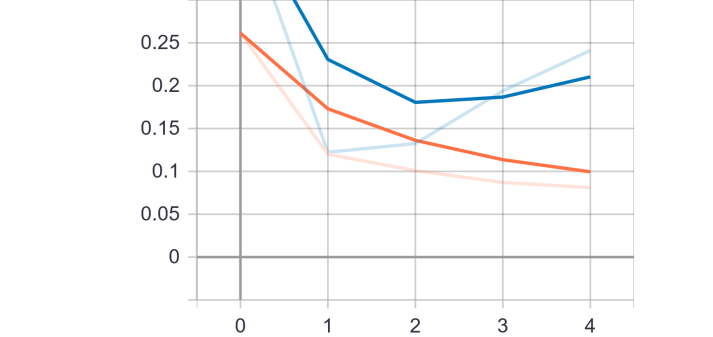
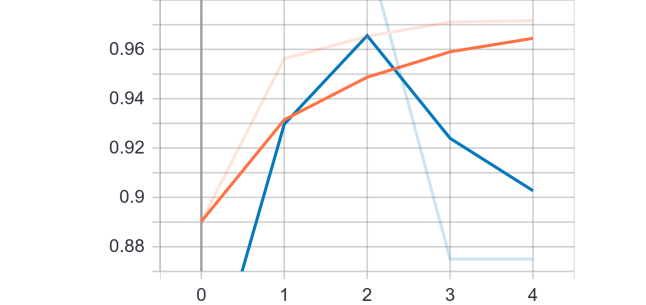


Precision Recall

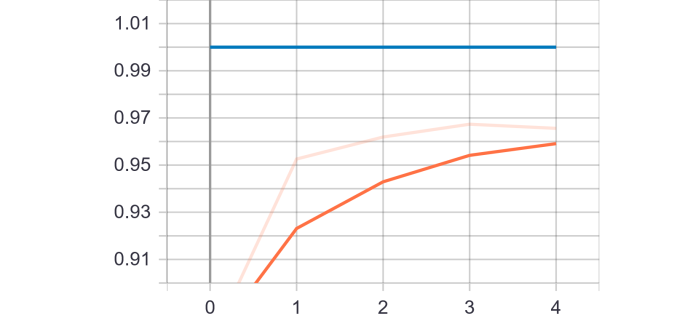
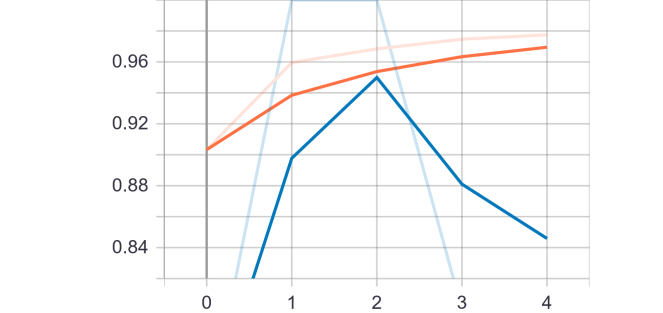


**Model 2 – SMOTEd data with 2 convolutions, 2 dropout layers, 2 fully connected layers**

Accuracy Loss

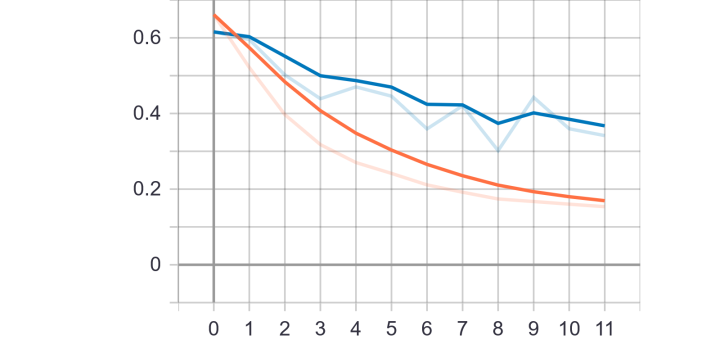
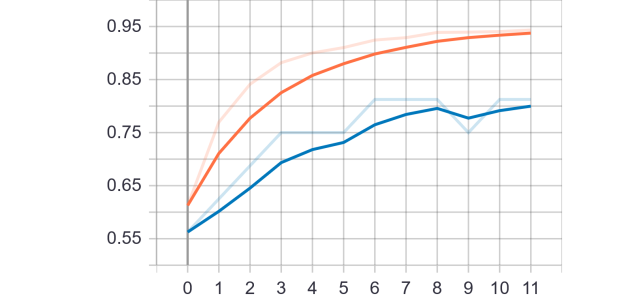


Precision Recall

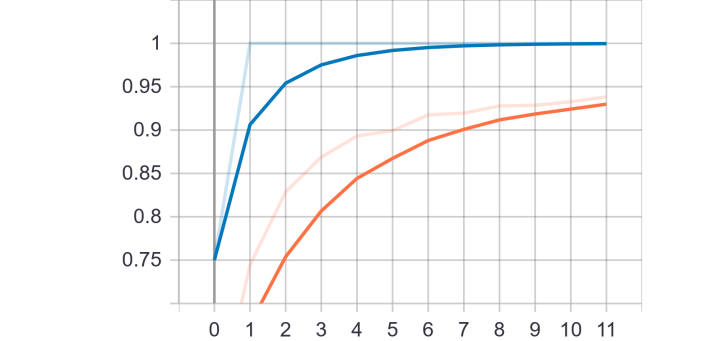
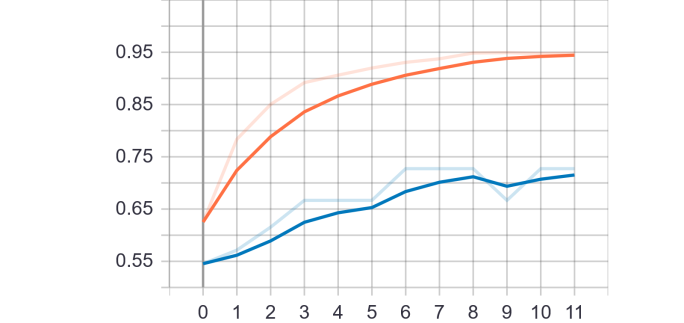


**Model 4 – SMOTEd data with 3 Convolutions, 2 Dropout layers, 3 Fully connected layers**

Accuracy Loss

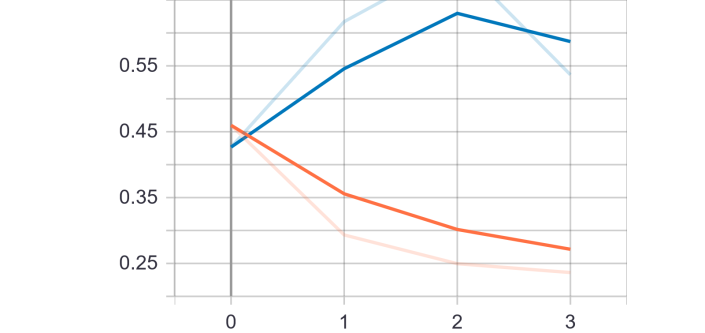
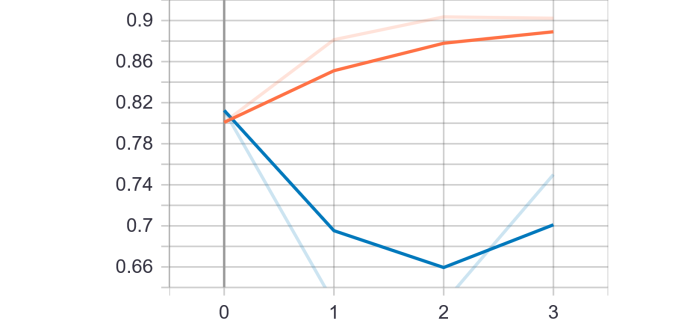


Precision Recall

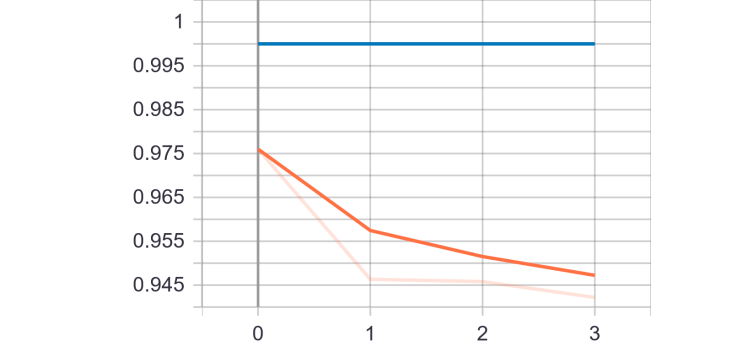
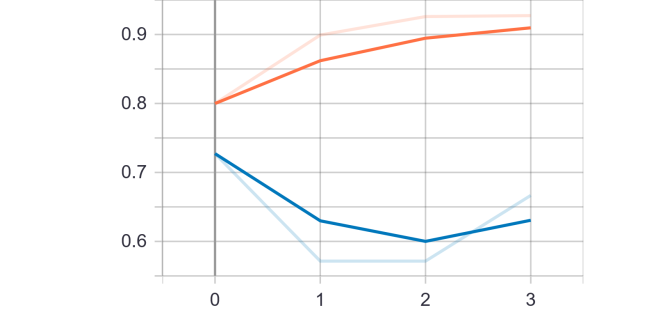


**Model 3 – NonSMOTEd data on Model 2**

Accuracy Loss

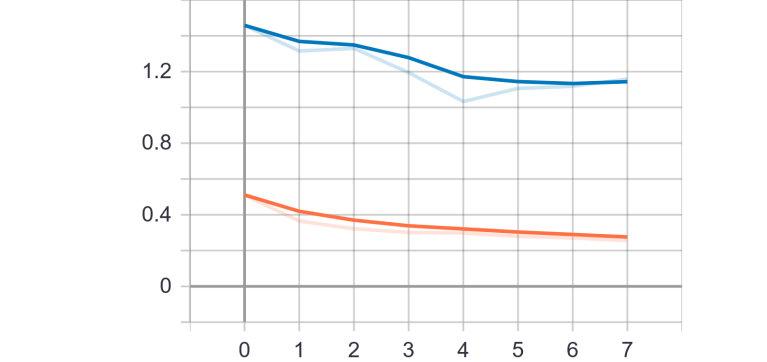
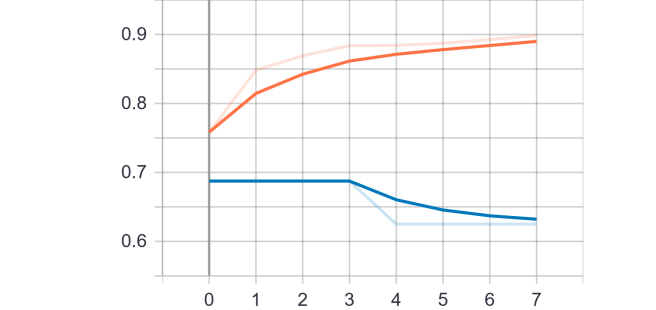


Precision Recall

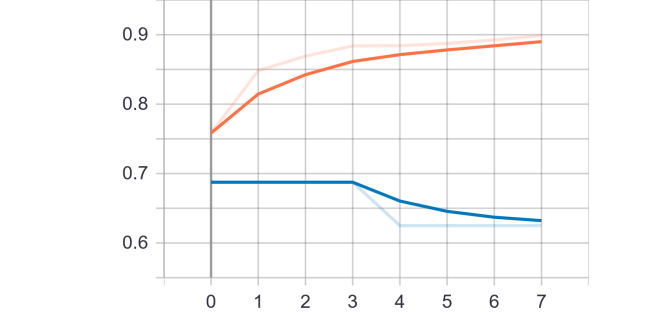
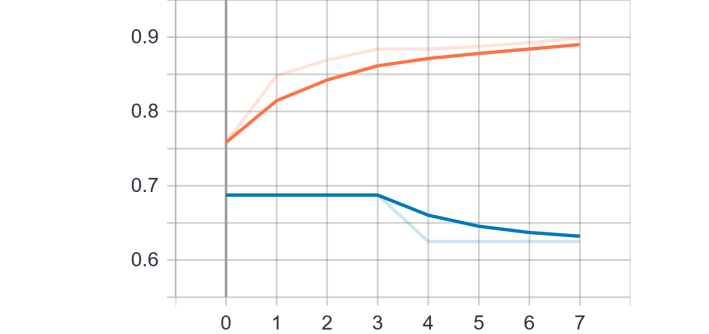


**Pretrained Model 1 – Inception V3 – Softmax with RMSprop**

Accuracy Loss

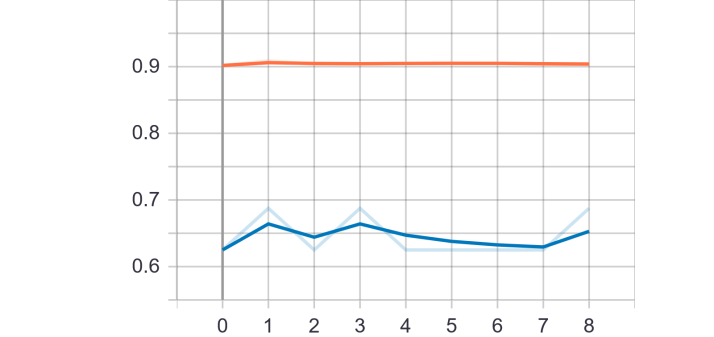
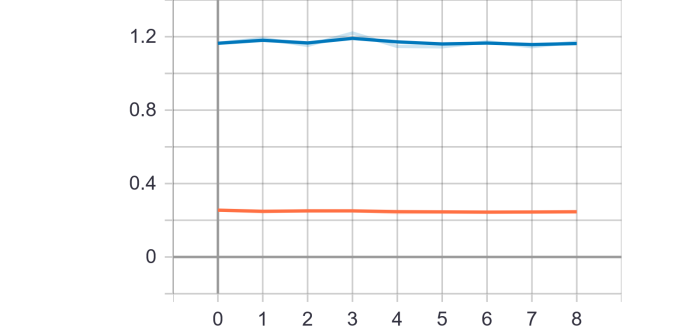


Precision Recall

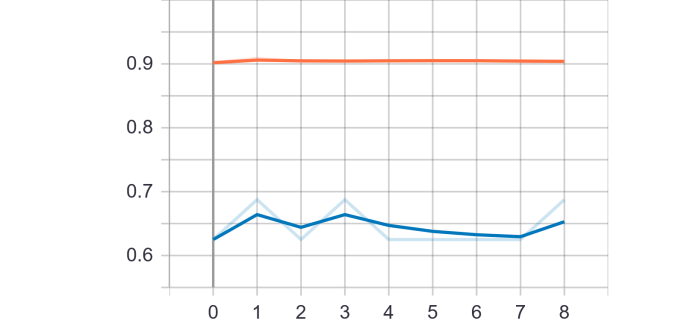
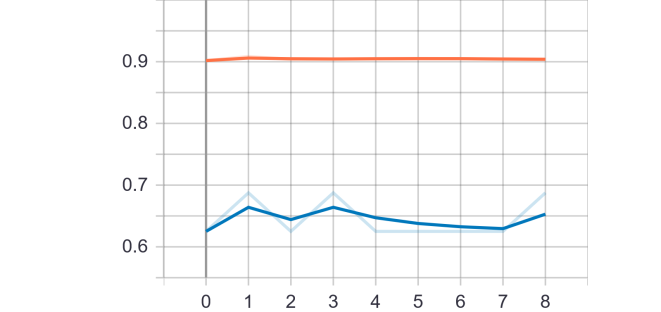


**Pretrained Model 2 – Inception V3 – Softmax with Adam**

Accuracy Loss

Precision Recall

**Comparison of Metrics across all models**







**Legends**

**S-M1, S-M2, S-M4**: SMOTEd data on 3 different CNN **NS-M3**: Non-SMOTEd data on M2

**S-T1/S-T2**: V3 Softmax RMSprop/Adam **S-T3/S-T4**: V3 Sigmoid RMSprop/Adam

**Summary**

The above figures summarize the results of this project. The first 3 models, S-M1, S-M2 and S-M4 are all variations of a convoluted neural network varying with the number of convolutions and dropout layers or fully connected layers. These 3 models used the SMOTEd balanced data set. The NS-M3 is the Non-SMOTEd data set using M2 architecture with class weights. The S-T1, S-T2, S-T3 and S-T4 are all transfer learning models with different output layers. T1 and T2 use Softmax activation with RMSprop and Adam optimizers respectively. T3 and T4 use Sigmoid activation with RMSprop and Adam optimizers.  
  
There is a slight increase in accuracy as we go from the 1st to the 3rd model in the graph. But, the NS-M3 has the highest accuracy and F1 Score among all. Of the 4 pre-trained models, the ones that used Softmax activation were the only one that showed some learning and performed reasonably well. The ones that used Sigmoid activation did not converge or did not learn anything.  
  
In this dataset, though the data is imbalanced, the positive or the Pneumonia cases are the majority and the normal or the negative cases are the minority. So, there is a bias to selecting the positive cases. This is reflected in the high recall but precision is low as compared to recall. Precision is also the most in the non-SMOTEd data set. The F1 score is also high in this model. So, Data augmentation with class weights seems to have worked better than SMOTE. Using Data Augmentation to produce data and then use it might be a better approach for imbalanced image datasets.   
  
Computationally, this project was run on an Ubuntu Intel Core i7-2760QM CPU @ 2.40GHz\*8 with NVC1 Graphics and 7.8 GB memory and 500 GB hard drive. Since, the dataset was not too big and could easily run the CNNs on this, I did not use the AWS service as I was having some difficulty/delay in getting my instance permitted.

According to the paper where this work was done using the pre trained model (InceptionV3), the model was run for close to 100 epochs. To achieve this and also try different optimizers or activations and different fine tuning methods for pre trained models, using AWS would be helpful.