**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**

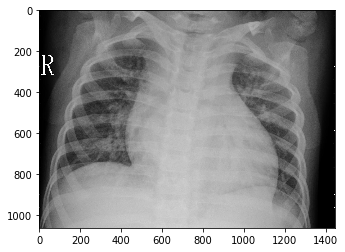
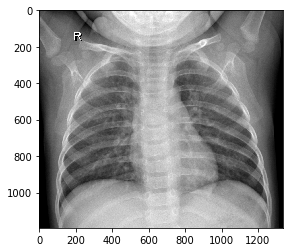
<http://www.cell.com/cell/fulltext/S0092-8674(18)30154-5>

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. I have only done rescaling and a horizontal flip to the image. The test set is only rescaled. This generator is used for the training set and the test set. The test set generator is used for the validation set too. This generator has 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 set is ready to be fit into the model for training it. We use the validation set for the validation step during training. We can evaluate model performance on the train or val set and then predict the test set.

**Model Architecture**

I am using a Sequential CNN model for this problem. 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 SoftMax activation. The model is compiled with the Adam optimizer, loss function is binary crossentropy and the metrics used to evaluate is Accuracy. As this is now a balanced data set, Accuracy is acceptable.

Below is the summary of the architecture and its shapes and parameters.

Model: "sequential\_12"

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

Layer (type) Output Shape Param #

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

conv2d\_27 (Conv2D) (None, 126, 126, 64) 1792

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

max\_pooling2d\_27 (MaxPooling (None, 63, 63, 64) 0

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

conv2d\_28 (Conv2D) (None, 61, 61, 32) 18464

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

max\_pooling2d\_28 (MaxPooling (None, 30, 30, 32) 0

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

conv2d\_29 (Conv2D) (None, 28, 28, 16) 4624

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

max\_pooling2d\_29 (MaxPooling (None, 14, 14, 16) 0

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

conv2d\_30 (Conv2D) (None, 12, 12, 8) 1160

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

max\_pooling2d\_30 (MaxPooling (None, 6, 6, 8) 0

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

conv2d\_31 (Conv2D) (None, 4, 4, 4) 292

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

max\_pooling2d\_31 (MaxPooling (None, 2, 2, 4) 0

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

flatten\_12 (Flatten) (None, 16) 0

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

dense\_26 (Dense) (None, 128) 2176

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

dense\_27 (Dense) (None, 64) 8256

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

dense\_28 (Dense) (None, 1) 65

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

Total params: 36,829

Trainable params: 36,829

Non-trainable params: 0