**PNEUMONIA DETECTION**

**CHEST X-RAY IMAGE CLASSIFICATION**

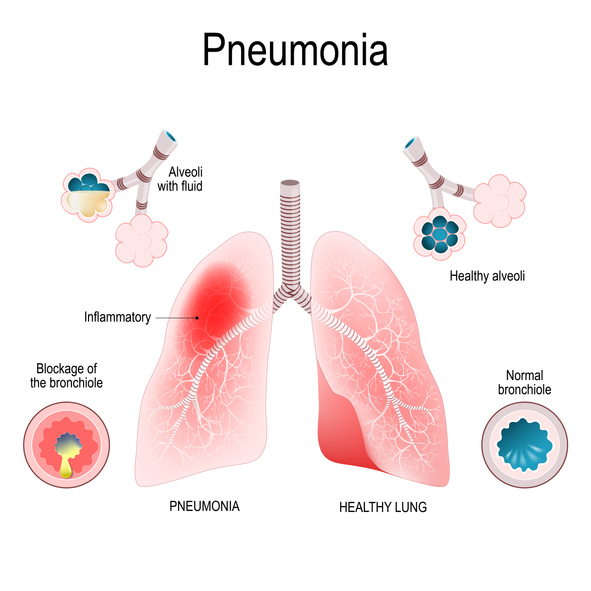
05-apr-2020

**VAIBHAV KUMAR**

# 

# **What is Pneumonia?**

Infection that inflames air sacs in one or both lungs, which may fill with fluid.With pneumonia, the air sacs may fill with fluid or pus. The infection can be life-threatening to anyone, but particularly to infants, children and people over 65.Symptoms include a cough with phlegm or pus, fever, chills and difficulty breathing.Antibiotics can treat many forms of pneumonia. Some forms of pneumonia can be prevented by vaccines.

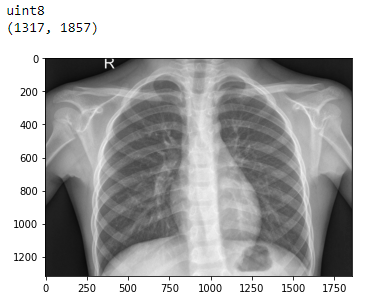


**Dataset Content**

We downloaded this dataset from kaggle. [Chest X-Ray Images (Pneumonia)](https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia)

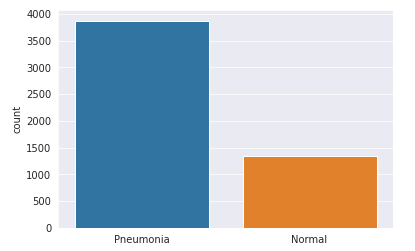
The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

**Randomly Visualising an X-Ray Image**



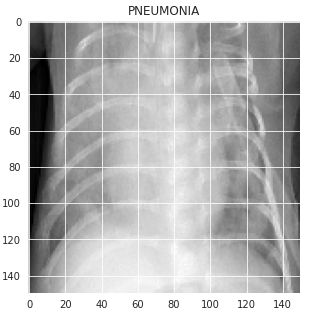
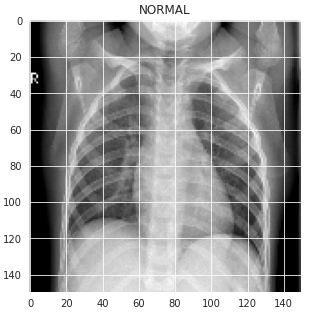
After this using os library and cv2 library we will make a loop and read all the images in the train, test and validation folder and make an array of all images and save them in ‘train’, ‘test’ and ‘val’ variables for further stages.

**Data Visualisation**



Bar graph of number of X-Ray images of Pneumonia and Normal class in Train folder. And from the above graph we can say that train data is imbalanced so later we will use Data Augmentation to increase the examples of training data.

**Visualising Both Classes Image**

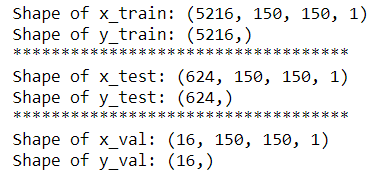
**Image Processing**

* After visualisation we will split the ‘train’, ‘test’ and ‘val’ into x and y, to form ‘x\_train’, ‘y\_train’, ‘x\_test’, ‘y\_test’, ‘x\_val’ and ‘y\_val’ where x represent the **Features** and y represent the **Labels**.
* After splitting we will perform **Grayscale Normalization** on ‘x\_train’, ‘x\_test’ and ‘x\_val’, i.e. all the images by dividing the np.array of all images with 255.
* After performing Grayscale Normalization we will **Reshape** the image of the data into (-1, 150,150,1) for model training.
* After all this process we will use **Data Augmentation** as we saw earlier our train data is imbalanced. For Data Augmentation we will use **ImageDataGenerator** from **keras.processing.image** library.

For Data Augmentation we used:

* Randomly rotate some training images by 30 degrees,
* Randomly Zoom by 20% some training images,
* Randomly shift images horizontally by 10% of the width and
* Randomly shift images vertically by 10% of the height.

Let’s see the Shape of ‘x\_train’, ‘y\_train’, ‘x\_test’, ‘y\_test’, ‘x\_val’ and ‘y\_val’:



**Model Training and Analysis**

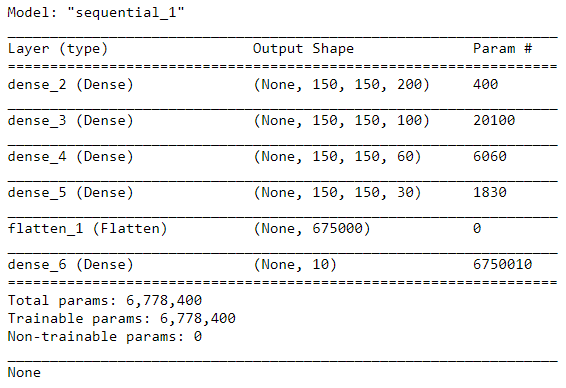
After Image processing lets train our model and do class prediction and model analysis.

For this Dataset we are gonna perform two models:

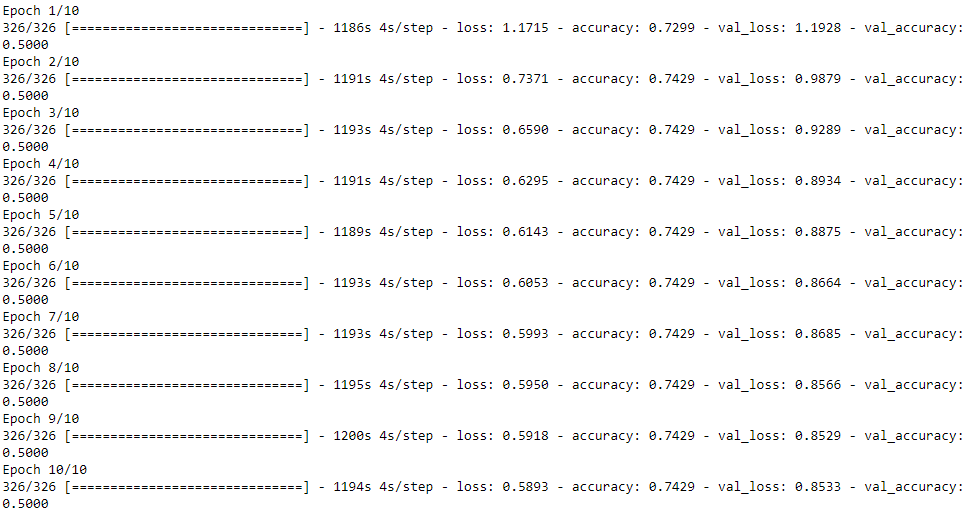
1. **Dense Neural Network and**
2. **Convolutional Neural Network**

* **Dense Neural Network**

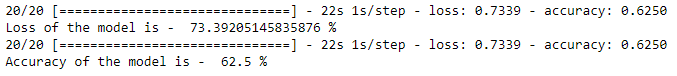
From **tensorflow.python** we import **Keras** and From Keras we import **Sequential**, **Dense**, **Flatten**, etc to configure our model of 5 layers of Dense with activation **‘relu’** and output activation of **‘softmax’**. And compile the model with **‘SGD’** optimizer with **‘sparse\_categorical\_crossentropy’** and **‘accuracy’** as metrics.

After compiling our model let’s fit our train and validation data into the model and run for 10 epochs.

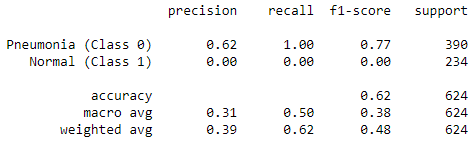
* DNN Model Performance



* Let’s check the loss and accuracy of the model

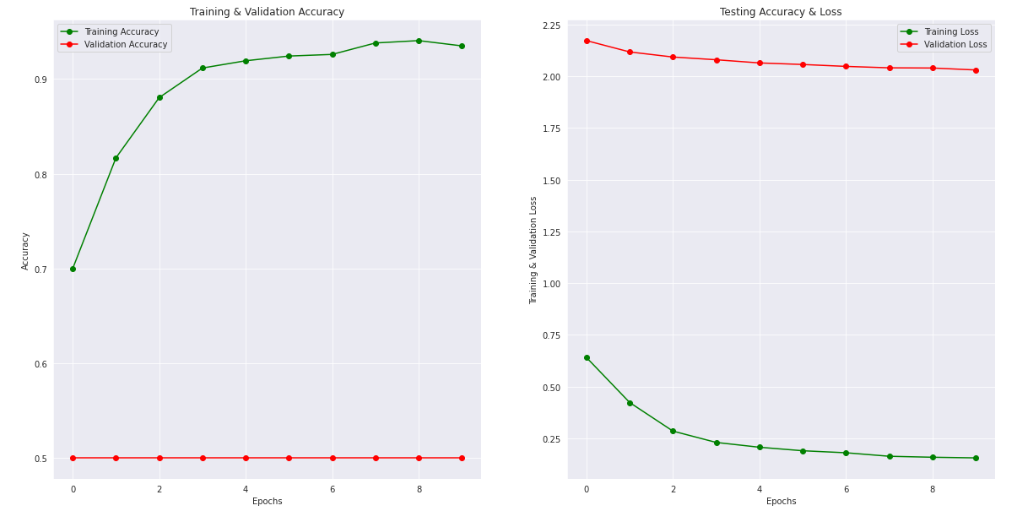


* Confusion Matrix of Model



* Model Analysis of DNN

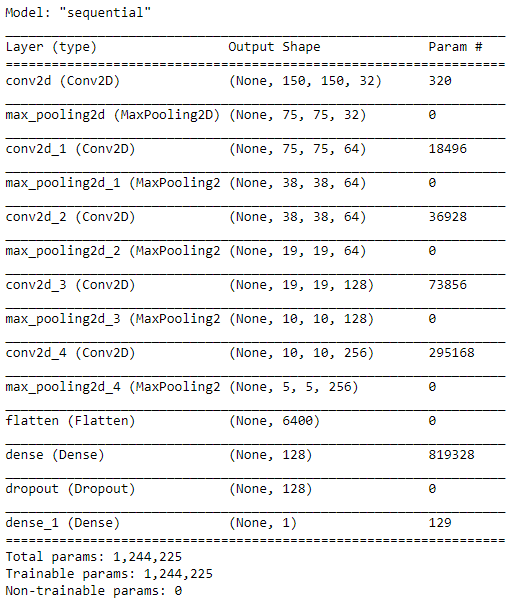
Line plot of Training Accuracy vs Validation Accuracy per epochs And Training Loss vs Validation Loss per epochs.



As we can see the confusion matrix classification report of precision, recall and f1 score of the normal class is 0.00 that shows our Dense Neural Network Model does not perform a Data Augmentation on the training set that's why it does classify the images well. So we will perform Convolutional Neural Network.

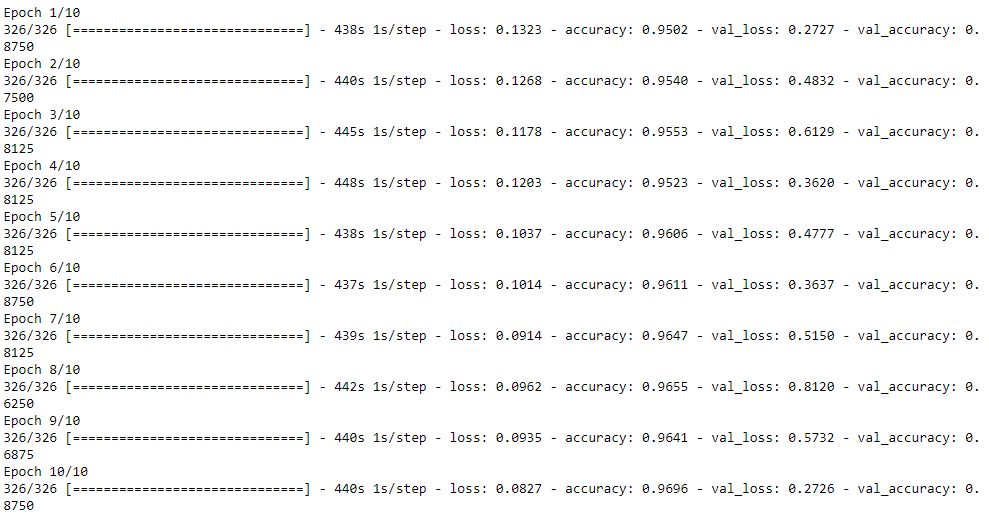
* Convolutional Neural Network

From **tensorflow.python** we import **Keras** and From Keras we import **Sequential**,**Conv2D**, **Maxpool2D**, **Dense**, **Flatten**, etc to configure our model of 10 layers of Conv2D with activation **‘relu’** and output activation of **‘sigmoid’**. And compile the model with **‘adam’** optimizer with **‘binary\_crossentropy’** and **‘accuracy’** as metrics.

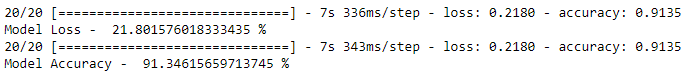


After compiling our model let’s fit our train and validation data into the model and run for 10 epochs.

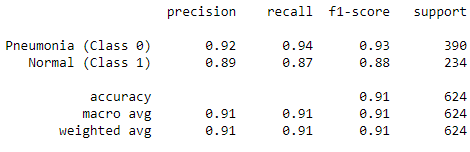
* CNN Model Performance



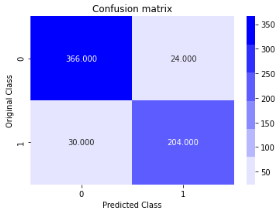
* Let’s check the loss and accuracy of the model



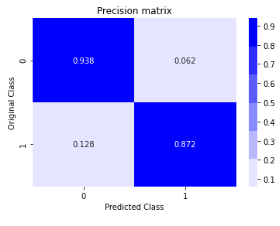
* Confusion Matrix of Model



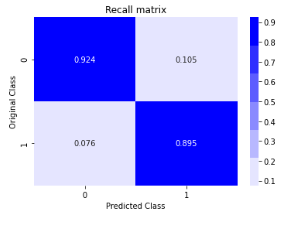
* Plotting Confusion, Precision and Recall Matrix
* Confusion Matrix



* Precision Matrix

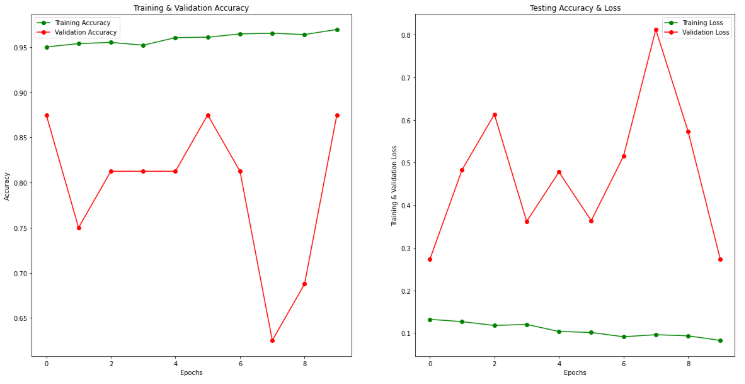


* Recall Matrix



* Model Analysis of CNN

Line plot of Training Accuracy vs Validation Accuracy per epochs And Training Loss vs Validation Loss per epochs.



After performing the model on our dataset we saw confusion matrix classification report, precision, recall and f1 score is almost balanced and model accuracy is 91%.