# Corona and pneumonia prediction from X-rays images using ResNet50

# Objectives:

The aim of this project is to detect COVID19, and PNEUMONIA from X-rays images by using ResNet50. COVID-19 (coronavirus disease 2019) is an infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), a strain of coronavirus. The first cases were seen in Wuhan, China, in late December 2019 before spreading globally. The current outbreak was officially recognized as a pandemic by the World Health Organization (WHO) on 11 March 2020. Currently Reverse transcription polymerase chain reaction (RT-PCR) is used for diagnosis of the COVID-19. X-ray machines are widely available and provide images for diagnosis quickly so chest X-ray images can be very useful in early diagnosis of COVID-19.

Dataset is organized into 2 folders (train, test) and both train and test contain 3 subfolders (COVID19, PNEUMONIA, NORMAL). Dataset contains total 6432 x-ray images and test data have 20% of total images. The ResNet50 model will be used to detect COVID19 & PNEUMONIA from the images.

# Methodology:

# Importing libraries:

The very first and most important step before implementing any machine learning and deep learning is to import all the required dependencies. I have imported Matplotlib library for the purpose of plotting the learning graph of my model. NumPy is imported to covert the X-rays images into numeric arrays. TensorFlow is imported to architect the ResNet50 model, the ResNet50 model come along with Keras.

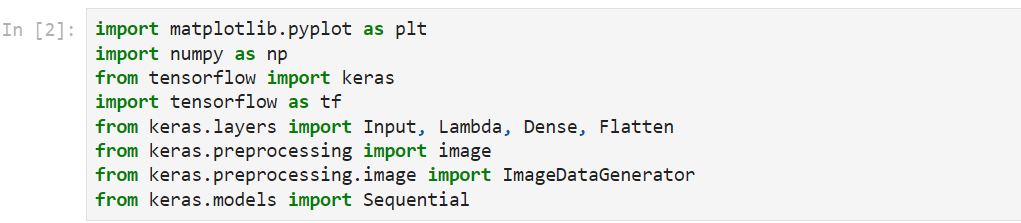


Figure 1 importing libraries

# Loading the data:

The dataset is classified into three different classes that is NORMAL, COVID19, and PNEUMONIA. We have the dataset containing images of chest X-rays, there are more than 6,000 images.

I have used a Python module called ‘SplitFolders’, which is used to split the dataset into Training, Testing, and Validation set, the dataset is split in the ratio of 70:20:1, means 80% data will be used for training, 20% for testing and the remaining 10% for validation purpose.

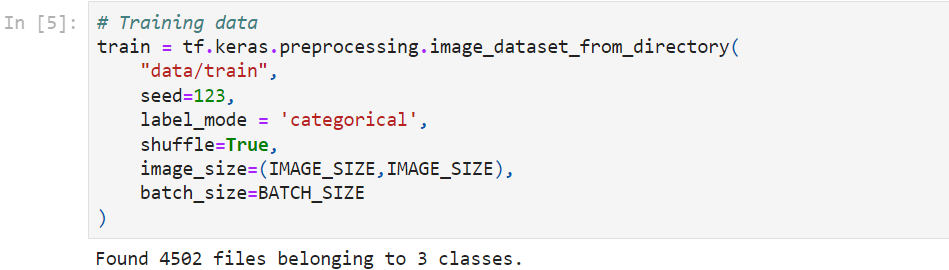


Figure 2 Loading data

In the above script, I have loaded the training data, the size of the images has been specified which is (*256,256*). In the same way, testing, and validation data have been uploaded.

# Model building:

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

I have uploaded the ResNet50 from keras application.

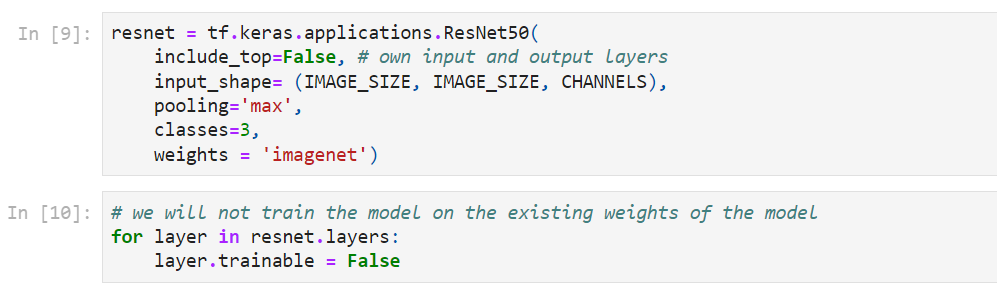


Figure 3 Loading ResNet50

First of all, the model has been imported from keras, the *include\_top=False* means that all the weights on which the model is trained on previously have been set to false, the model will not use those weights to train on our dataset, we will add our own hidden layer. The input shape of the images has been given to the model and I have used the MaxPooling for the pooling purpose from the filters. The number of classes is 3 because our output will be one out of three classes. And the weights are specified as imagenet because the ResNet50 is already trained on the imagenet database.

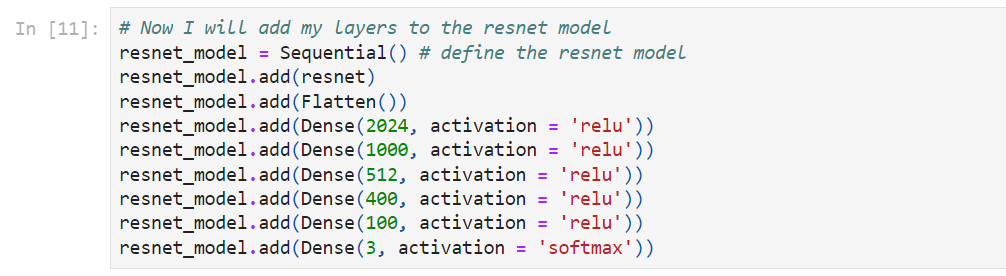


Figure 4 Adding hidden layers

After importing the model, I have added six more hidden layers at the end of the ResNet50 architecture, these layers will take my images as an input and perform all the operations and will give me the results at the last layer. In the first layer, I have used 2024 hidden neurons and the activation function is relu, in the end, the neurons are 3 because the output of my model will be three classes. After that, the model is compiled by using the Adam optimizer and loss as categorical cross entropy, and metrics as accuracy.

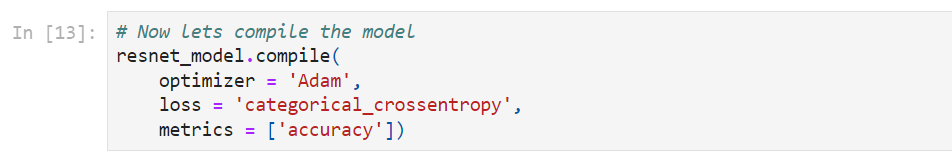


Figure 5 Compiling

# Results:

After Training the model, I got almost 100% accuracy on training data, and 95.66% accuracy on validation dataset.

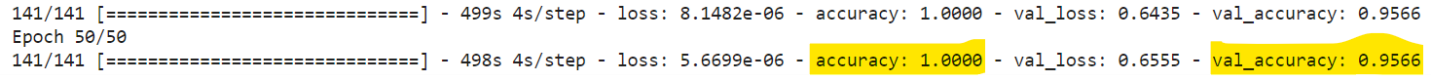


Figure 6 training and validation accuracy

On testing data, I got the accuracy of 96.96%.

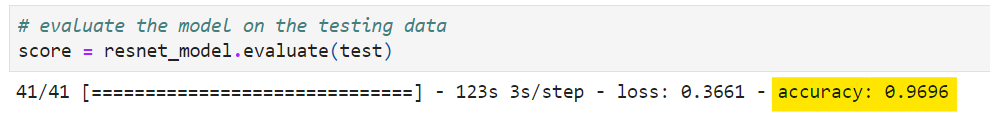


Figure 7 Testing Accuracy

The model performed really very well.

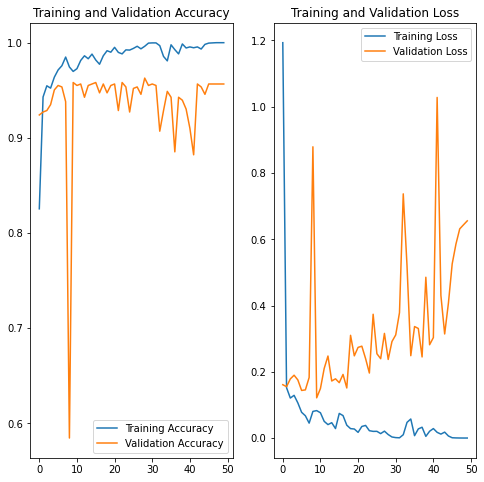


Figure 8 Training and Validation accuracy

After training the mode and evaluated it on the testing data, now it is the time to perform some predictions on the model using the testing dataset, these testing images are completely unseen by our model, The predictions on the testing images is shown below.

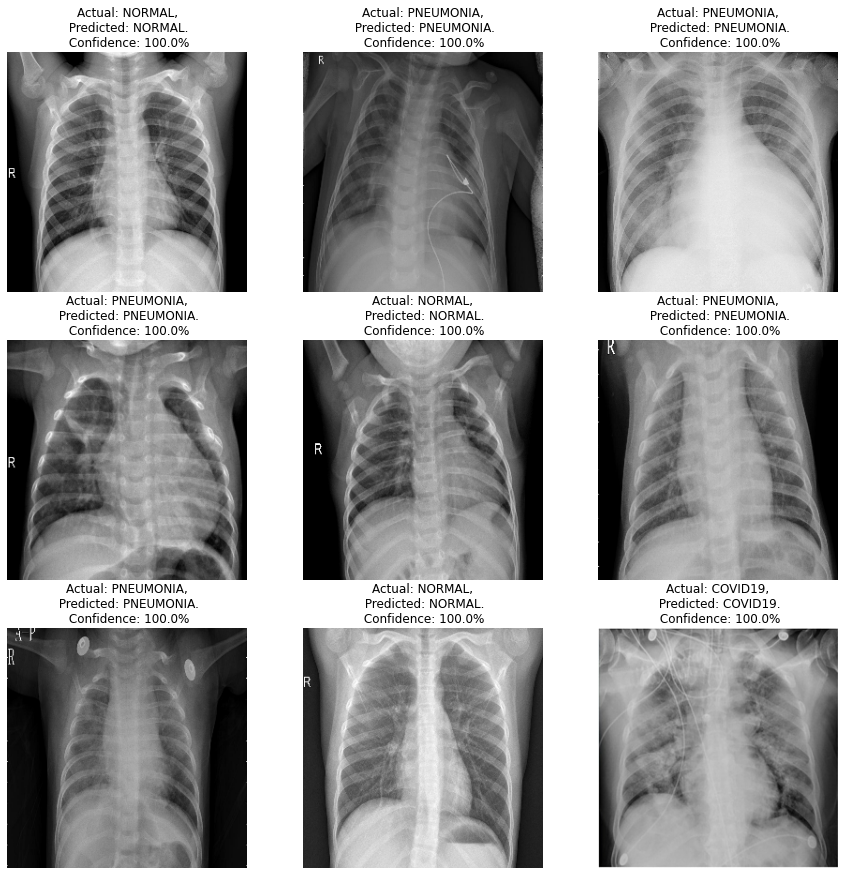


Figure 9 Predictions

We can see the actual label and predicted label along with the confidence of the model about each prediction.