Pneumonia Covid Normal Fever Detection Using CT Scan Images

Name: Manish Kumar Rohilla **Email:** kotamanishrohilla@gmail.com
Registration No. : 11714083 **Phone No:** 8750606272

Abstract

In this project we will use the effect of deep convolutional network on its accuracy in the classification and recognition of large-scale images. We will use the Pretrained Neural Network VGG16 for its state of the art feature extraction properties. The extracted features will be saved in CSV format file to train on Support Vector Machine Algorithm Library of sklearn. Using pretrained model as a feature extraction layer by freezing complete model is a very efficient way to implement project on system with low hardware specs, as we dont need to train the model again and model training utilizes most of the harware capability. Although I have used SVM from sklearn and XGBoost model in my project but any other model which can carry out classification or basic regression can also be used. In this project we will also try to tune hyperparameters on our classification model to know the best parmeters to get the best results. Model saving will aso be done to make sure that we dont need to carry out the whole process again and again.

Introduction

Convolutional Networks or ConvNets are very important part of every model when we talk about image or video recognition or classification. In particular, an important role in the advancement of deep visual recognition architectures has been played by Deep Convulational Network based models. Some of the best examples of pretrained models that is known for their accuracy in Image recognition are ImageNets, MobileNet-V2, Inception-V3, GoogLeNet, VGG16, VGG19 ,etc. We can directly import these models and use the layers we want and can further tune it by adding more layers to improve it. The main problem that a person can face during using the pretrained models are that in case the data you are going to feed to the model is disimilar to the data on which the model is trained then there is a good chance that the extracted features will not be of great use and accuracy will be less. So it is better u first examine the models pros and cons before using it and then build a model of your own. Throughout this project we will use various functions from 2 very important machine learning libraries of python, those libraries are Tensorflow and Sklearn.
Apart from that Numpy and Pandas will be used to handle the structures that will store the vectorized image data.

Technology Used

For Data Handling:

Numpy: It is one of the core library for scientific computing which provides a highly efficient multidimensional array with some useful tools for manipultiong and creating those arrays.

Pandas: It is a software library used for data manipulation and analysis, it have data structures known as dataframe along with various useful functions to manipulate or create data. This very library will also be used for loading features form CSV files in to dataframes for further training of Machie Learning Algorithms.

For Data Visualization:

Matplotlib: This library gives us a way to represent our data in a visual form. You can plot your data on any graph accordingly to gain insight knowledge of the distribution and classification with overall similarities and difference.

For Feature Extraction:

Tensorflow: It is an end to end open source platform for Machine Learning developed by Google. This library provides the feature of adding any number of layers in your model without actually typing code for it. And when we talk about deep learning with thousands of layers densely connected to each other it will be one heck of a problem to build even a single model without Tensorflow.

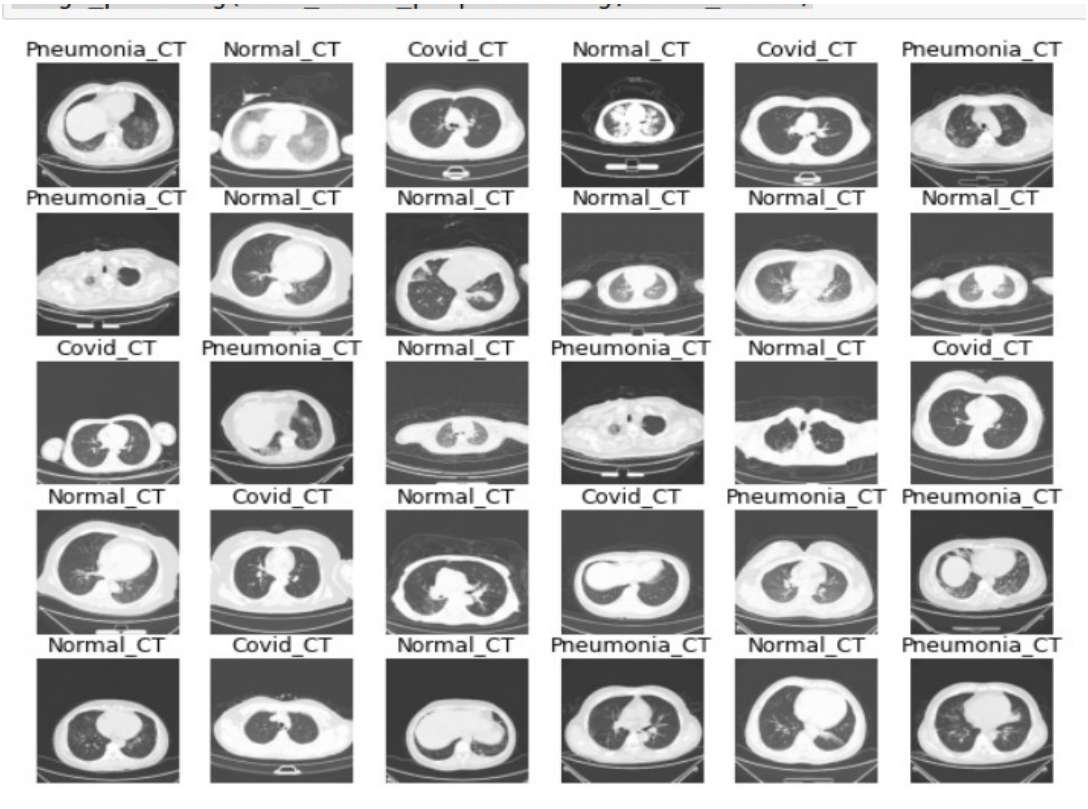
Keras: Keras is one of the leading high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines. Keras API required for working with TensorFlow computational graphs, tensors, operations, and sessions.

Dataset

Dataset is taken from the following link:
https://www.kaggle.com/anaselmasry/covid19normalpneumonia-ct-images

The dataset consists of 3 files naming:
1: Covid2 CT
2: Normal CT
3: Pneumonia CT

There are total approx. 2034 Images in the dataset of size 324MB in total.

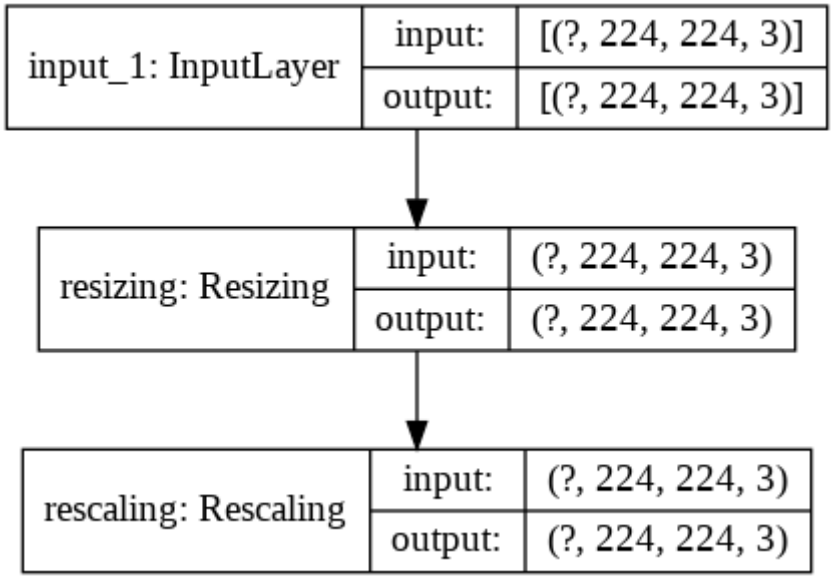


Data Preprocessing:

For converting the data into a format that is trainable by the Model for feature Extraction, we will use Tensorflow library.

Command:

```
tf.keras.preprocessing.image_dataset_from_directory(
    location,
    image_size=image_size,
    batch_size=batch_size)
    return data
```

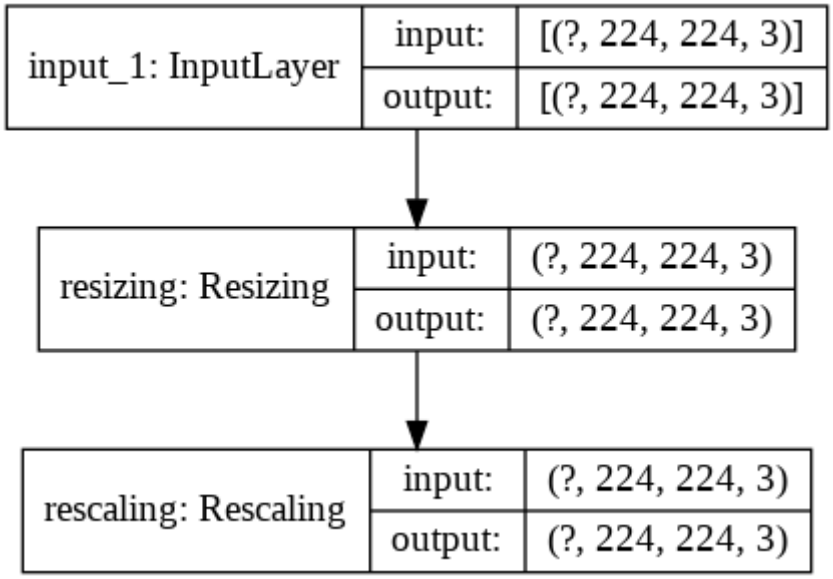


After that we will use RandomScaling and Resizing on our data to further clean our data and Normalize it. These steps will give us the final dat feedable to the model further

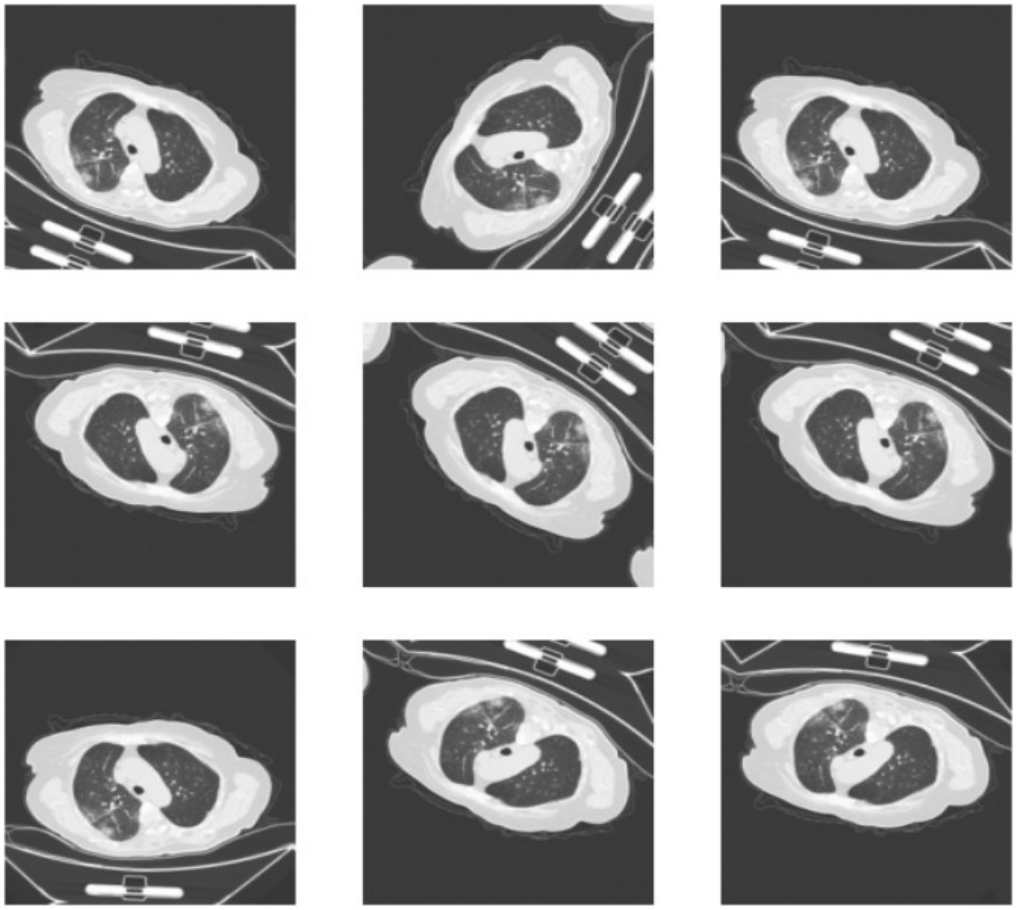
Note: We will use image size as (224,224) because the pretrained model VGG16 I used for feature extraction use Images of this size.

Data Augmentation:

Data Augmentation creates the diversity in the data we are feeding to our model for training. This helps our model ready to face some complications while predicting the outcome of certain input in which the provided image data is manipulated. The manipulation of image data comes in the sense of rotating the Input Image horizontally or vertically, or rescaling it. The image data can be maniulated in various ways like croping,rotating,padding, etc. So to prevent these manipulations during prediction Data Augmentation is necessary. It is also used to avoid overfitting of our model.



Example of How Data Augmentation works on One Image:



MODEL

As we are going to use VGG16, we need to import it and turn top layer of VGG16 to False and make the whole model Untrainable. So that we can use the previously stored weights and parameters to extract features from it.

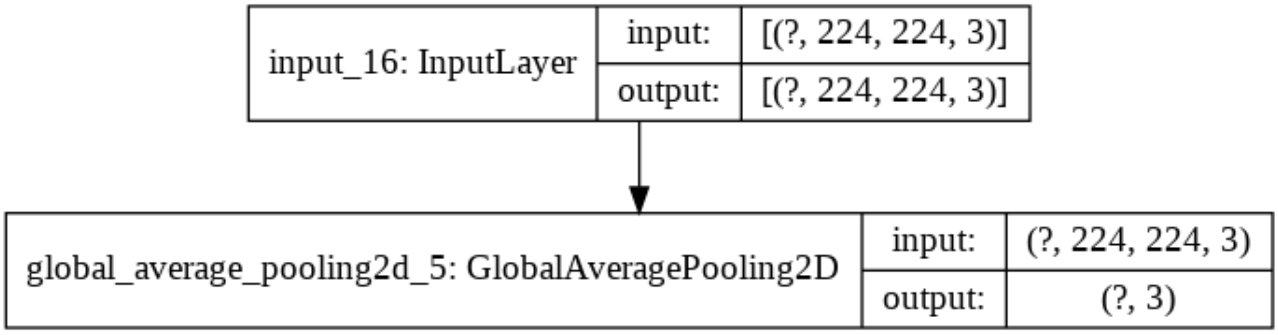
The VGG16 model summay after removing the top layer will look like :

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_15 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,740,338
Trainable params: 14,740,338
Non-trainable params: 0
```

NOTE: After making the model Non Trainable ,in summary we will observe the **Non Trainable Parameters are 14,714,688 out of Total 14,740,338**. The **trainable parameters will be 25,650** which belongs from the layers we are going to add further in VGG16.

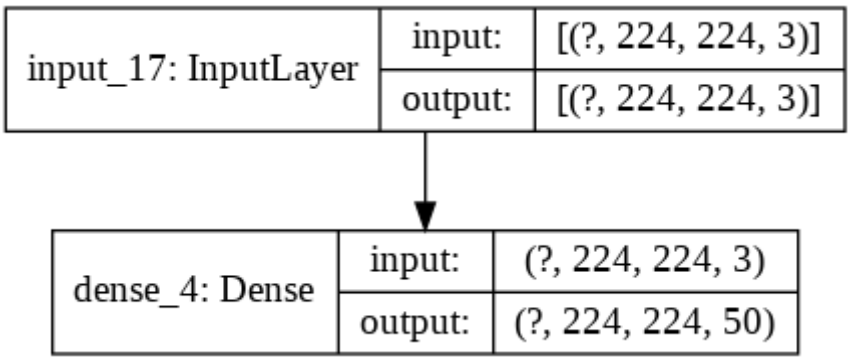
GLOBAL AVERAGE POOLING:

Our data is in spatial matrix form so further we will compute the average value of all values across the entire input width and input height matrix for every channel.

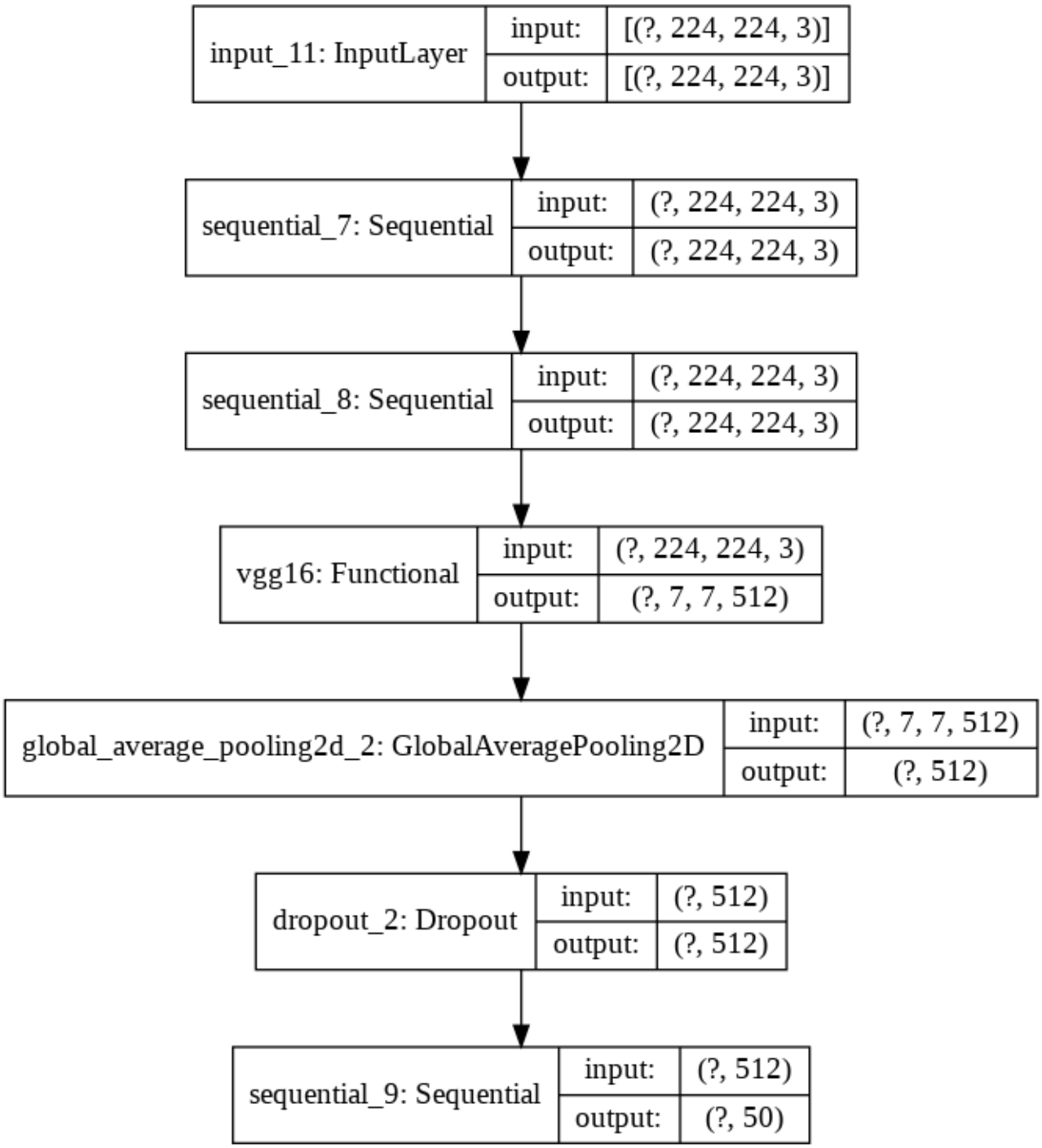


Dense Layer on Output

This layer will give the output features according to the number of argument passed during adding this layer to our model. In my case I passed 50 as argument, So I will receive 50 features as outputs. Basically this layer gives output in the form of proability.



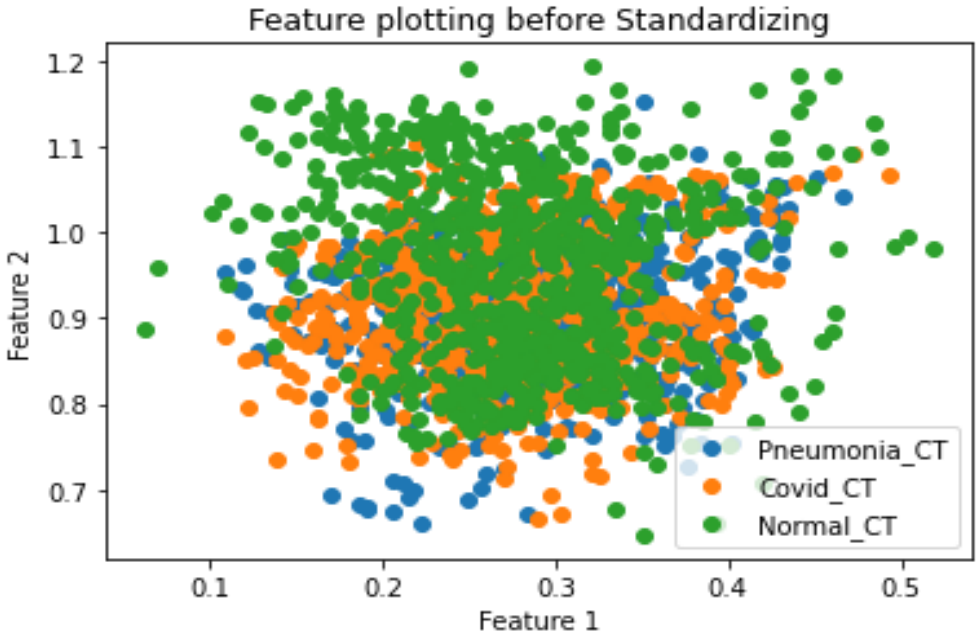
Final Model Schema:

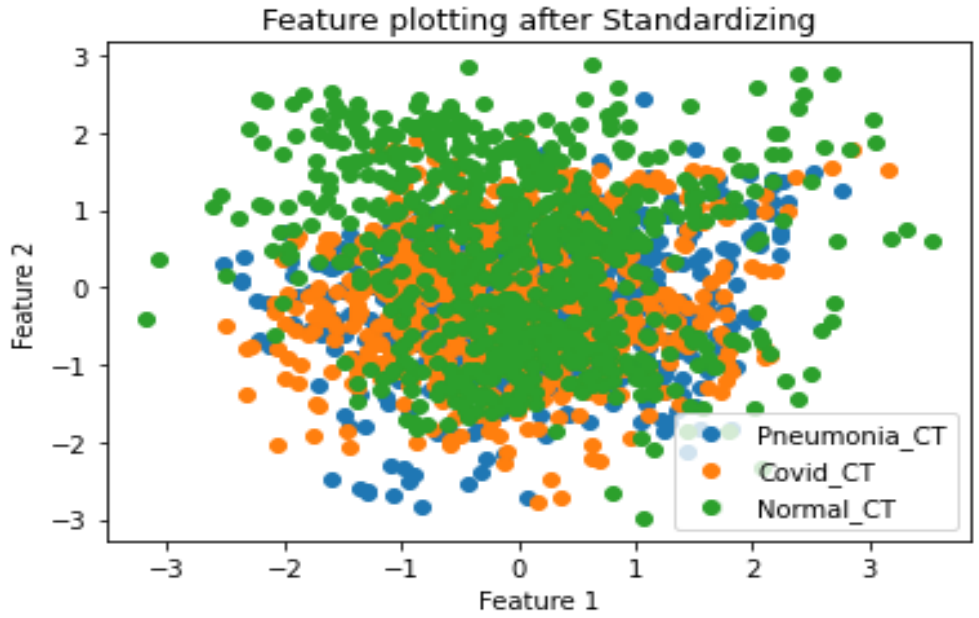


Feature Saving and Plotting:

The feature we extracted with the help of our model we will save them in CSV format using Pandas As we took 50 features from each image and there are total of 2034 images the feature output will be of shape 2034 rows and 50 columns. The labels will be of shape 2034 rows only with 1 column.

We will separate the data on the basis of labels and will plot it before Standardizing the data and after Standardizing the data.





SVM (Support Vector Machine) Machine Learning Algorithm

Support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.
In sklearn we can import SVM from SVC which suports various kernel for training like rbf (Radial Basis Functions) , linear (Linear Regression), poly (Polynomial Regression).
We will use Polynomial Kernel in our model for training.
After training we will use testing data to check the prediction accuracy of our model.

Conclusion of SVM Model
Results of our model are as follows:

Intercepts: [0.14550914 0.35647554 0.25066951]
R2 Score of Training: 1.0
R2 Score of Testing: 0.9921414538310412

Mean Absolute Error for Training: 0.0
Mean Squared Error for Training: 0.0
Root mean squared error for Training: 0.0

Mean Absolute Error for Testing: 0.011787819253438114
Mean Squared Error for Testing: 0.019646365422396856
Root mean squared error for Testing: 0.14016549298025122

Training accuracy: 1.0
Testing accuracy: 0.9921414538310412

Cross Validation Scores for 5 folds

Scoring on Negative Mean Squared Error
Scores: [0.11451967 0.19835388 0.11451967 0.30835364 0. **]**
Mean: 0.147
Std: 0.102

GitHub Link: https://github.com/ManishRohilla/Pneumonia_Covid_Normal_Fever_CT