Configuration Manual

MSc Research Project
Programme Name

Abhijeet Anand
Student ID: X19216068

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Abhijeet Anand
Student ID:	x19216068
Programme:	Masters in Data Analytics **Year:** 2020-2021
Module:	Research Project
Lecturer:	Vladimir Milosavljevic
Submission Due Date:	16th December 2021
Project Title:	Automated CAD System for Classification of Covid-19 using Xception Model
Word Count:	………1546…………………………………………………………**Page Count:** …16…

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Abhijeet Anand
……

Date: 31st January 2022
……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Automated CAD System for Classification of Chest X-Rays using Xception Model

Abhijeet Anand
Student ID:x19216068

1 Introduction

This document is used to explain all the system requirements and the coding interpretation briefly. It would also give the walkthrough of all the code execution, tools and libraries used and the CAD system designed. It would briefly explain how to use pretrained model and analyses the performance of model using the various metrices like recall, accuracy, precision, F1 score and the confusion matrix. At last, it would show how to save the model and use it in developing a web-based application using Flask.

2 Configuration of the System

This section is used to brief about the system configuration of the local machine used in the implementation of research. While executing the Deep learning projects system specifications plays and important role in efficiency and overall performance of the model. In this work both local and virtual environment has been used:

2.1 Physical Machine Configuration:

The Physical Machine is used to run the Google Colab and Anaconda on it.

1. Operating System: Windows 10 64-bit
2. Processor: Intel i7 10th Generation
3. Hard Disk: 512 GB SSD
4. Memory: 8 GB of RAM

2.2 Tools & Library used:

1. **Google Colab**: Colab is also known as Colaboratory, it facilitates to run the code in python using a browser. It is mostly used for the machine learning and data analytics.

2. **Google Drive**: It is a service deployed by Google in the year 2012 to store files in synchronized way. It provides Cloud storage to synchronize and share files on various platform and devices. The cloud is encrypted and proactively secured with malware and phishing detection activated on them.

3. **Anaconda Navigator**: It is GUI based python distribution that is used to launch common applications like jupyter Notebook, Spyder, etc. It is used for common python programs without using the command line to install conda packages. It can also search for new updates in local repository or on Anaconda.org.

4. **Flask:** It is a lightweight web framework which is coded in python for API designing. It is sometime known as micro web framework because it gives a lot of features without particular tools and libraries.

5. **Spyder IDE**: It is an open-source development platform used python programing. It is used to execute the app.py code pass the images uploaded using html form into model.h5 file.

6. **NumPy**: It is python library which is installed to work on multidimensional matrices and arrays. It consists of many mathematical functions which helps in proper calculation in neural network.

7. **Pandas**: It is a python library used for data analysis and manipulation. It consists of some operations and data structures.

8. **TensorFlow**: It is a software library used in python to perform tasks in machine learning AI. It is already installed in Google colab for training of deep neural networks.

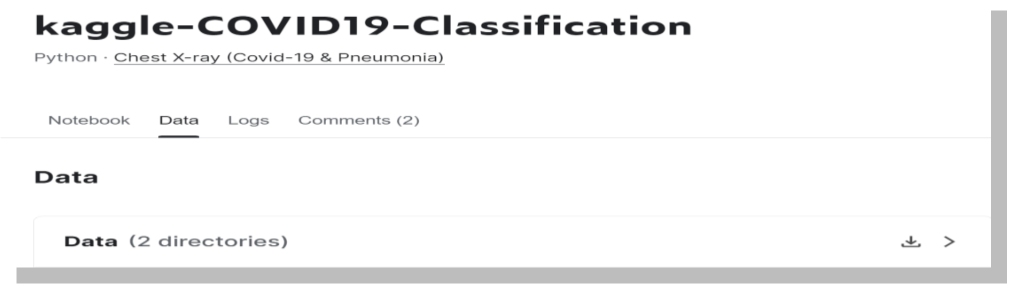
9. **Keras**: It is a python library used as a interface in TensorFlow and Aritifical Network. It is used for neural network layers, activation functions, objectives, optimizers and other tools which are necessary for working with image data.

10. **Scikit-Learn**: It is a python library used for statistical modeling and machine learning which includes clustering, classification and regression.

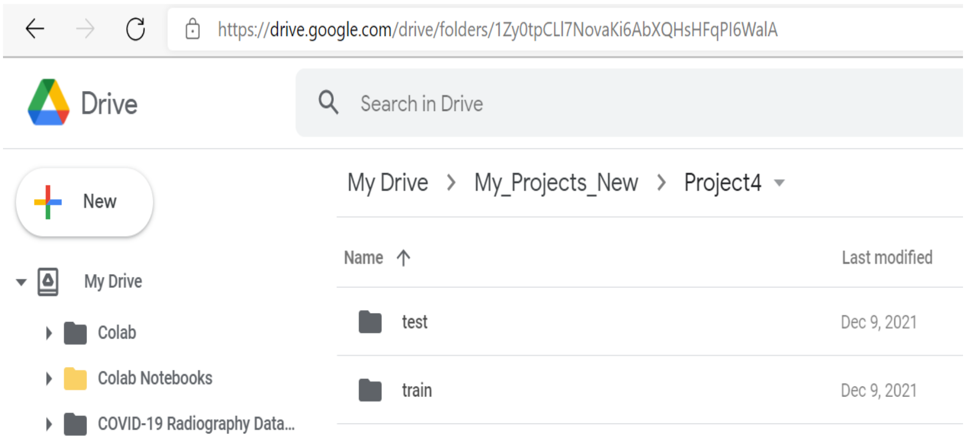
11. **HDF5 File**: HDF5 stands for Hierarchical Data Format, it supports big and heterogenous data. It consists of a structured directory which helps in keeping data and file in structured and organized way. It is used for storing the trained model in this project.

3 Implementation:

1. Download the dataset from the Kaggle website. The dataset contains two folders train and test. (https://www.kaggle.com/bimsarananayakkara/kaggle-covid19-classification/data)



2. Uploaded the dataset on the Google drive.



3. Open the Google Colab using the browser and **mount** the Google drive. It would verify the access using a verification link.

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

4. Import the important Python Libraries like keras, numpy, Pandas, matplotlib and tensorflow.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import datetime
import tensorflow as tf
import keras
from keras import models
from keras import layers
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dense, Conv2D , SeparableConv2D, MaxPool2D , Flatten ,Dropout , BatchNormalization,
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from keras.callbacks import ReduceLROnPlateau
from keras import backend as K
from keras import optimizers
from sklearn.metrics import classification_report, recall_score, precision_score, confusion_matrix, f1_score,
```

5. Load the images from train folder and encode the classes as numerical 0,1 and 2.

```
path = '/content/drive/MyDrive/My_Projects_New/Project4/train'

diag_code_dict = {
    'COVID19': 0,
    'NORMAL': 1,
    'PNEUMONIA': 2}
```

6. Load the image data into test and train path. **ImageDataGenerator** from keras library is a class used to for data augmentation in real-time and generate images at every epoch. **Flow_from_directory** is a method of Imagedatagenerator class used to take the path of directory and generate augmented data.

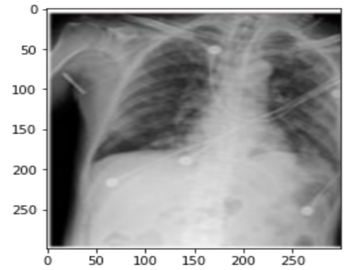
```
test_path = '/content/drive/MyDrive/My_Projects_New/Project4/test/'
train_path = '/content/drive/MyDrive/My_Projects_New/Project4/train/'

classes = ["COVID19", "NORMAL", "PNEUMONIA"]
num_classes = len(classes)

#Training data
train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=10,
        zoom_range=0.4,
        horizontal_flip=True,
        validation_split=0.01
        )

train_generator = train_datagen.flow_from_directory(
        '/content/drive/MyDrive/My_Projects_New/Project4/train',
        target_size=(299, 299),
        batch_size=32,
        class_mode='categorical',
        subset='training'
        )
#validation data
val_generator = train_datagen.flow_from_directory(
        '/content/drive/MyDrive/My_Projects_New/Project4/train',
        target_size=(299, 299),
        batch_size=32,
        class_mode='categorical',
        subset='validation')
#testing data
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
        '/content/drive/MyDrive/My_Projects_New/Project4/test',
        target_size=(299, 299),
        batch_size=32,
        class_mode='categorical')
```

```
Found 5094 images belonging to 3 classes.
Found 50 images belonging to 3 classes.
Found 1288 images belonging to 3 classes.
```

7. Load a sample image of Covid-19 case from the database.

```
from keras.preprocessing import image
#visualize data
image_path = "/content/drive/MyDrive/COVID-19 Radiography Database/COVID-19/COVID-19 (123).png"
new_img = image.load_img(image_path, target_size=(299, 299))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
print("COVID-19")
plt.imshow(new_img)
```

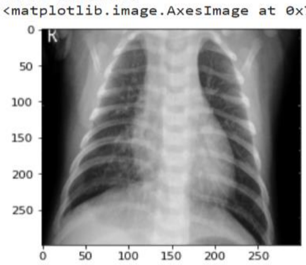
```
COVID-19
<matplotlib.image.AxesImage at 0x7f95e4f42210>
```



8. Load a sample image of Viral Pneumonia case from the database.

```
#visualize data
image_path = "/content/drive/MyDrive/COVID-19 Radiography Database/Viral Pneumonia/Viral Pneumonia (1024).png"
new_img = image.load_img(image_path, target_size=(299, 299))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
print("Viral Pneumonia")
plt.imshow(new_img)
```

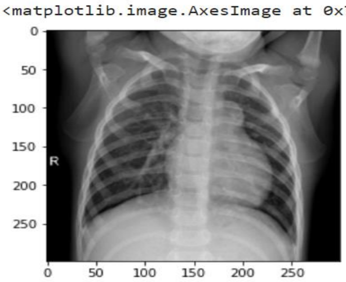
```
Viral Pneumonia
<matplotlib.image.AxesImage at 0x7f95e4a9c450>
```



9. Load a sample image of Normal case from the database.

```
#visualize data
image_path = "/content/drive/MyDrive/COVID-19 Radiography Database/NORMAL/NORMAL (1013).png"
new_img = image.load_img(image_path, target_size=(299, 299))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
print("NORMAL")
plt.imshow(new_img)
```

```
NORMAL
<matplotlib.image.AxesImage at 0x7f95e4a1a450>
```



10. Instantiate the Xception architecture using pre-trained weights of ImageNet. (Adusumilli, 2020)(f.keras.applications.xception.Xception).

```
#Using pretrained Xception model
xception=tf.keras.applications.Xception(input_shape=(299,299,3),
                                        include_top=False,
                                        weights='imagenet')

#freezing the layers
for layer in xception.layers:
    layer.trainable=False
```

11. Relu function is used in the hidden layer and Softmax layer to classify the images in a multiclass classification whereas Sigmoid is used in binary classification in logistic regression.(S, 2021)

```
x=layers.Flatten()(last_output)

#adding an extra layer
x=layers.Dense(256,activation='relu')(x)

#output layer
x=layers.Dense(3,activation='softmax')(x)

xception_model=keras.Model(xception.input,x)
```

12. Configure the model using **model.compile().**

```
xception_model.compile(optimizer='adam',
                       loss='categorical_crossentropy',
                        metrics=['accuracy'])
```

13. How the model looks like showing the 14 blocks and number of parameters which are trainable and non-trainable.

```
xception_model.summary()

Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 299, 299, 3  0           []
                                )]

 block1_conv1 (Conv2D)          (None, 149, 149, 32  864         ['input_1[0][0]']
                                )

 block1_conv1_bn (BatchNormaliz  (None, 149, 149, 32  128        ['block1_conv1[0][0]']
 ation)                         )

 block1_conv1_act (Activation)  (None, 149, 149, 32  0           ['block1_conv1_bn[0][0]']
                                )

 block1_conv2 (Conv2D)          (None, 147, 147, 64  18432       ['block1_conv1_act[0][0]']
                                )

 block1_conv2_bn (BatchNormaliz  (None, 147, 147, 64  256        ['block1_conv2[0][0]']
 ation)                         )

 block1_conv2_act (Activation)  (None, 147, 147, 64  0           ['block1_conv2_bn[0][0]']
```

```
block2_sepconv1 (SeparableConv  (None, 147, 147, 12  8768       ['block1_conv2_act[0][0]']
2D)                            8)

block2_sepconv1_bn (BatchNorma  (None, 147, 147, 12  512        ['block2_sepconv1[0][0]']
lization)                      8)

block2_sepconv2_act (Activatio  (None, 147, 147, 12  0          ['block2_sepconv1_bn[0][0]']
n)                             8)

block2_sepconv2 (SeparableConv  (None, 147, 147, 12  17536      ['block2_sepconv2_act[0][0]']
2D)                            8)

block2_sepconv2_bn (BatchNorma  (None, 147, 147, 12  512        ['block2_sepconv2[0][0]']
lization)                      8)

conv2d (Conv2D)                (None, 74, 74, 128)  8192       ['block1_conv2_act[0][0]']

block2_pool (MaxPooling2D)     (None, 74, 74, 128)  0          ['block2_sepconv2_bn[0][0]']

batch_normalization (BatchNorm  (None, 74, 74, 128)  512        ['conv2d[0][0]']
alization)
block14_sepconv1_act (Activati  (None, 10, 10, 1536  0          ['block14_sepconv1_bn[0][0]']
on)                            )

block14_sepconv2 (SeparableCon  (None, 10, 10, 2048  3159552    ['block14_sepconv1_act[0][0]']
v2D)                           )

block14_sepconv2_bn (BatchNorm  (None, 10, 10, 2048  8192       ['block14_sepconv2[0][0]']
alization)                     )

block14_sepconv2_act (Activati  (None, 10, 10, 2048  0          ['block14_sepconv2_bn[0][0]']
on)                            )

flatten (Flatten)              (None, 204800)       0          ['block14_sepconv2_act[0][0]']

dense (Dense)                  (None, 256)          52429056   ['flatten[0][0]']

dense_1 (Dense)                (None, 3)            771        ['dense[0][0]']

==================================================================================================
Total params: 73,291,307
Trainable params: 52,429,827
Non-trainable params: 20,861,480
__________________________________________________________________________________________________
```

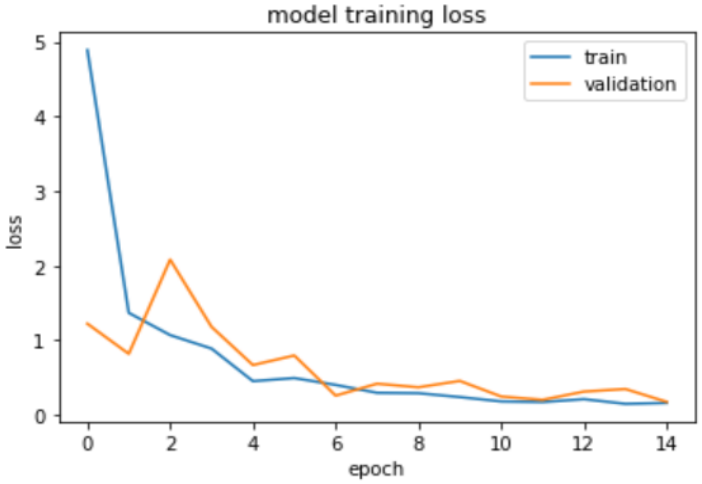
14. Train the model using Xception_model.fit**()**

```
history_xception=xception_model.fit(train_generator,
                                    validation_data=val_generator, steps_per_epoch= 40,epochs=15)
```

```
Epoch 1/15
40/40 [==============================] - 539s 13s/step - loss: 4.8889 - accuracy: 0.8211 - val_loss: 1.2215 - v
Epoch 2/15
40/40 [==============================] - 501s 12s/step - loss: 1.3667 - accuracy: 0.8852 - val_loss: 0.8170 - v
Epoch 3/15
40/40 [==============================] - 492s 12s/step - loss: 1.0682 - accuracy: 0.8900 - val_loss: 2.0808 - v
Epoch 4/15
40/40 [==============================] - 500s 12s/step - loss: 0.8879 - accuracy: 0.8961 - val_loss: 1.1789 - v
Epoch 5/15
40/40 [==============================] - 497s 12s/step - loss: 0.4511 - accuracy: 0.9180 - val_loss: 0.6667 - v
Epoch 6/15
40/40 [==============================] - 495s 12s/step - loss: 0.4924 - accuracy: 0.9062 - val_loss: 0.7951 - v
Epoch 7/15
40/40 [==============================] - 501s 12s/step - loss: 0.4006 - accuracy: 0.9133 - val_loss: 0.2570 - v
Epoch 8/15
40/40 [==============================] - 499s 12s/step - loss: 0.2949 - accuracy: 0.9305 - val_loss: 0.4168 - v
Epoch 9/15
40/40 [==============================] - 496s 12s/step - loss: 0.2893 - accuracy: 0.9250 - val_loss: 0.3689 - v
Epoch 10/15
40/40 [==============================] - 503s 13s/step - loss: 0.2376 - accuracy: 0.9141 - val_loss: 0.4542 - v
Epoch 11/15
40/40 [==============================] - 503s 13s/step - loss: 0.1791 - accuracy: 0.9391 - val_loss: 0.2450 - v
Epoch 12/15
40/40 [==============================] - 485s 12s/step - loss: 0.1713 - accuracy: 0.9330 - val_loss: 0.2007 - v
Epoch 13/15
40/40 [==============================] - 508s 13s/step - loss: 0.2096 - accuracy: 0.9258 - val_loss: 0.3112 - v
Epoch 14/15
40/40 [==============================] - 504s 13s/step - loss: 0.1472 - accuracy: 0.9406 - val_loss: 0.3458 - v
Epoch 15/15
40/40 [==============================] - 502s 13s/step - loss: 0.1589 - accuracy: 0.9406 - val_loss: 0.1733 - v
```

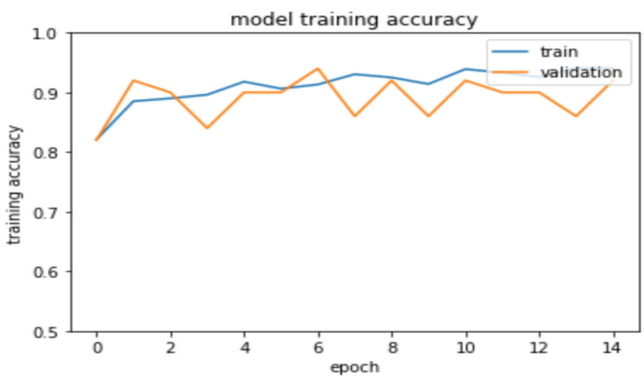
15. **Train and Validation loss Graph**: It shows the model is learning and both loss are gradually decreasing, with validation loss less than training loss is good for model. A model is said to be underfitting if it has high training and validation error. A model is said to be over fitting if it has low training error and high validation error.

```
plt.plot(history_xception.history['loss'])
plt.plot(history_xception.history['val_loss'])
plt.title('model training loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
plt.show()
```



16. **Train and Validation Accuracy:** Both Train and Test accuracy graph are at higher values showing that model is good. Also, we can see that model is gradually learning.

```
plt.plot(history_xception.history['accuracy'])
plt.plot(history_xception.history['val_accuracy'])
plt.title('model training accuracy')
plt.ylabel('training accuracy')
plt.xlabel('epoch')
plt.ylim([0.5,1])
plt.legend(['train', 'validation'], loc='upper right')
plt.show()
```



17. Evaluate the model on test data and check its loss and accuracy.

```
#Evaluate the model on test data and check its loss and accuracy.
x=xception_model.evaluate(test_generator)
```

```
41/41 [==============================] - 441s 11s/step - loss: 0.2571 - accuracy: 0.9123
```

```
print(f'Testing loss: {x[0]}')
print(f'Testing accuracy: {x[1]}')
```

```
Testing loss: 0.25709158182144165
Testing accuracy: 0.9122670888900757
```

18. Predict the class of the image

```
#Predict the class(Covid,Viral Pneumonia or Normal) of each image
y_predictions=[]
for img in os.listdir("/content/drive/MyDrive/My_Projects_New/Project4/test/COVID19"):
    img = load_img('/content/drive/MyDrive/My_Projects_New/Project4/test/COVID19/'+img,target_size=(299,299))
    img = img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    result = xception_model.predict(img/255.0)[0]
    y_predictions.append(np.argmax(result))

for img in os.listdir("/content/drive/MyDrive/My_Projects_New/Project4/test/NORMAL"):
    img = load_img('/content/drive/MyDrive/My_Projects_New/Project4/test/NORMAL/'+img,target_size=(299,299))
    img = img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    result = xception_model.predict(img/255.0)[0]
    y_predictions.append(np.argmax(result))

for img in os.listdir("/content/drive/MyDrive/My_Projects_New/Project4/test/PNEUMONIA"):
    img = load_img('/content/drive/MyDrive/My_Projects_New/Project4/test/PNEUMONIA/'+img,target_size=(299,299)
    img = img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    result = xception_model.predict(img/255.0)[0]
    y_predictions.append(np.argmax(result))

y_predictions=np.array(y_predictions)
```

19. Evaluation of model based on various parameters precision, recall and F1

```
#View the Class indices of the model
test_generator.class_indices
```

```
{'COVID19': 0, 'NORMAL': 1, 'PNEUMONIA': 2}
```

```
#Evaluation metrices for model precision, recall, F1- score
print(classification_report(test_generator.classes,y_predictions))
```

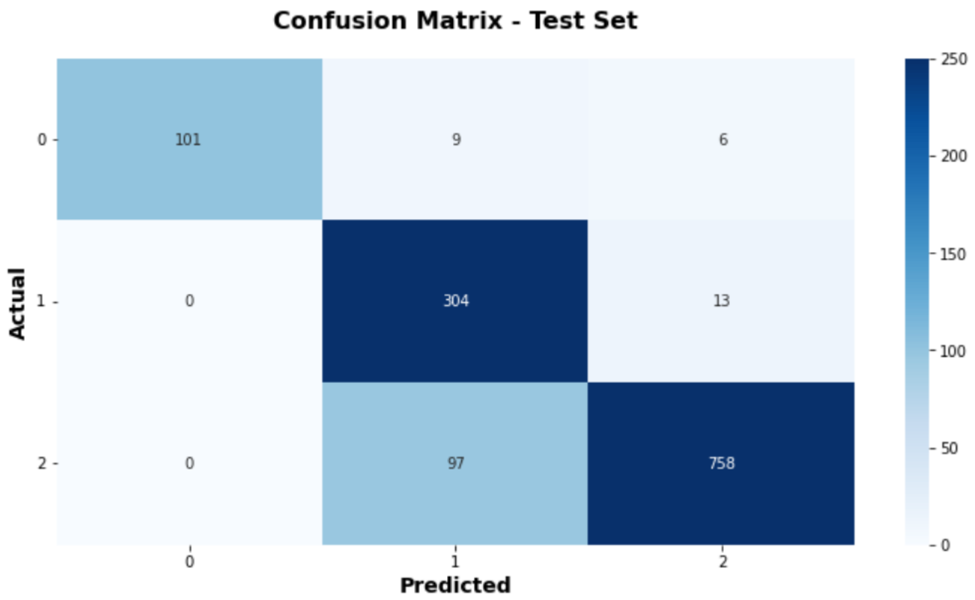
```
              precision    recall  f1-score   support

           0       1.00      0.87      0.93       116
           1       0.74      0.96      0.84       317
           2       0.98      0.89      0.93       855

    accuracy                           0.90      1288
   macro avg       0.91      0.91      0.90      1288
weighted avg       0.92      0.90      0.91      1288
```

20. Print confusion matrix to check the number of True positives, True Negative, False positive and False Negative values.

```
print(confusion_matrix(test_generator.classes,y_predictions))
```

```
[[101   9   6]
 [  0 304  13]
 [  0  97 758]]
```

```
import seaborn as sns


plt.figure(figsize=(12, 6))
ax = sns.heatmap(confusion_matrix(test_generator.classes,y_predictions), annot = True, fmt = 'g' ,vmin = 0, vma
ax.set_xlabel('Predicted',fontsize = 14,weight = 'bold')
ax.set_xticklabels(ax.get_xticklabels(),rotation =0);

ax.set_ylabel('Actual',fontsize = 14,weight = 'bold')
ax.set_yticklabels(ax.get_yticklabels(),rotation =0);
ax.set_title('Confusion Matrix - Test Set',fontsize = 16,weight = 'bold',pad=20);
```



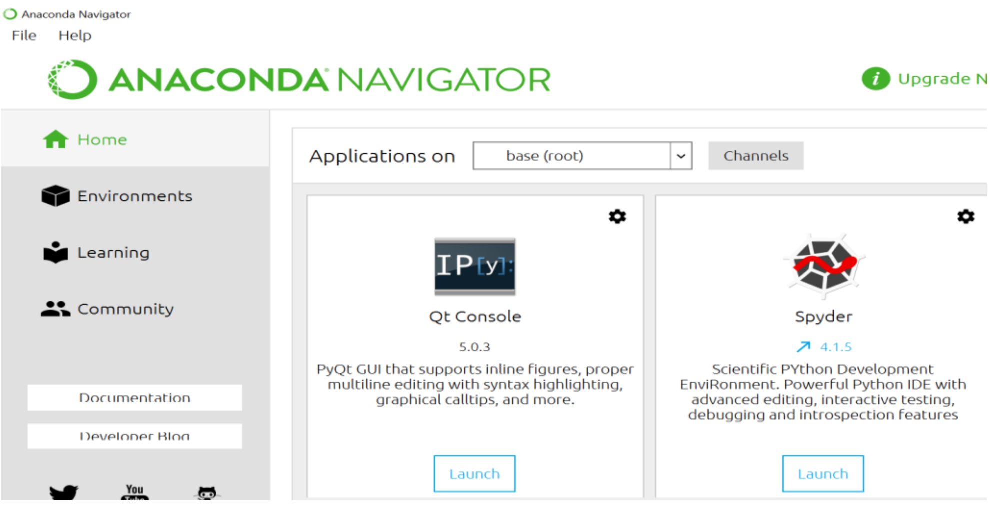
21. Print graph for recall, precision and F1-score

```
acc = accuracy_score(test_generator.classes,y_predictions)
from tensorflow.keras.metrics import PrecisionAtRecall,Recall
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_recall_fscore_support, accuracy_score
results_all = precision_recall_fscore_support(test_generator.classes,y_predictions, average='macro', zero_divi
results_class = precision_recall_fscore_support(test_generator.classes,y_predictions, average=None, zero_divis

metric_columns = ['Precision','Recall', 'F1-Score','S']
all_df = pd.concat([pd.DataFrame(list(results_class)).T,pd.DataFrame(list(results_all)).T])
all_df.columns = metric_columns
all_df.index = ['COVID','Normal', 'Viral Pneumonia','Total']

def metrics_plot(df,metric):
    plt.figure(figsize=(22,10))
    ax = sns.barplot(data =df, x=df.index, y = metric,palette = "Blues_d")
    #Bar Labels
    for p in ax.patches:
        ax.annotate("%.1f%%" % (100*p.get_height()), (p.get_x() + p.get_width() / 2., abs(p.get_height())),
        ha='center', va='bottom', color='black', xytext=(-3, 5),rotation = 'horizontal',textcoords='offset poi
    sns.despine(top=True, right=True, left=True, bottom=False)
    ax.set_xlabel('Class',fontsize = 14,weight = 'bold')
    ax.set_ylabel(metric,fontsize = 14,weight = 'bold')
    ax.set(yticklabels=[])
    ax.axes.get_yaxis().set_visible(False)
    plt.title(metric+ ' Results per Class', fontsize = 16,weight = 'bold');

metrics_plot(all_df, 'Precision')#Results by Class
metrics_plot(all_df, 'Recall')#Results by Class
metrics_plot(all_df, 'F1-Score')#Results by Class
```

22. Save the trained mode as .h5 file to use it in flask.

```
xception_model.save('xception_model.h5')

/usr/local/lib/python3.7/dist-packages/keras/engine/functional.py:1410: CustomMaskWarning: Custom mask layers
  layer_config = serialize_layer_fn(layer)
```

23. Install the Anaconda Navigator on the local system. Once Anaconda is installed open the Anaconda Navigator and Lunch Spyder 4.1.5 IDE on it.



24. Browse the path of folder where the python file for the flask integration is stored.

25. Open the app.py file and install the necessary libraries. (Flask and Keras in this case)

...:\Users\Abhijeet Anand\Downloads\Thesis Documents\Thesis_FinalCode&Data\Code\FlaskLocalSpyder\app.py

app.py*

```
# In[6]:

from flask import Flask, render_template, request
from keras.models import load_model
from keras.preprocessing import image
import numpy as np

```

26. App is started using Flask(_name_) and dictionary is created a dictionary with classes Covid :0, Normal :1 and Viral Pneumonia:2 (Ankit, 2021). Then the weight of trained model is loaded into model.

```
app = Flask(__name__)

dic = {0 : 'COVID-19', 1 : 'NORMAL', 2 : 'Viral Pneumonia' }

model = load_model('xception_model.h5')

model.make_predict_function()
```

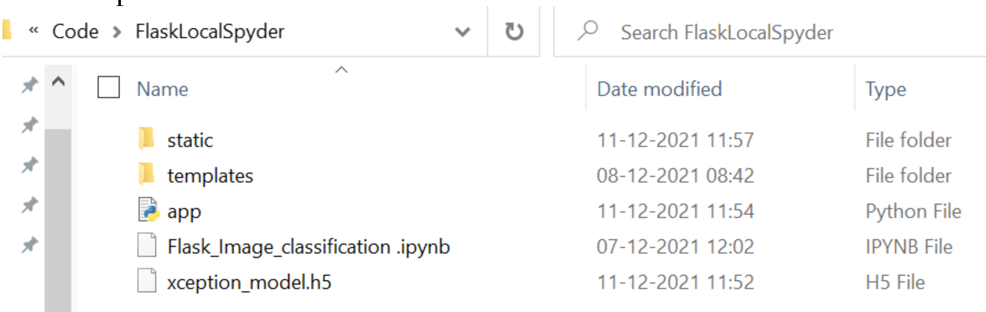
27. Predict function is used to take the images from the html form and them reshape then into size which passes into model. i.e. 299x299 and then return the predicted value.

```
def predict_label(img_path):
    new_img = image.load_img(img_path, target_size=(299,299))
    img = image.img_to_array(new_img)
    img = np.expand_dims(img, axis = 0)
    p = np.argmax(model.predict(img/255.0), axis=1)
    return dic[p[0]]
```

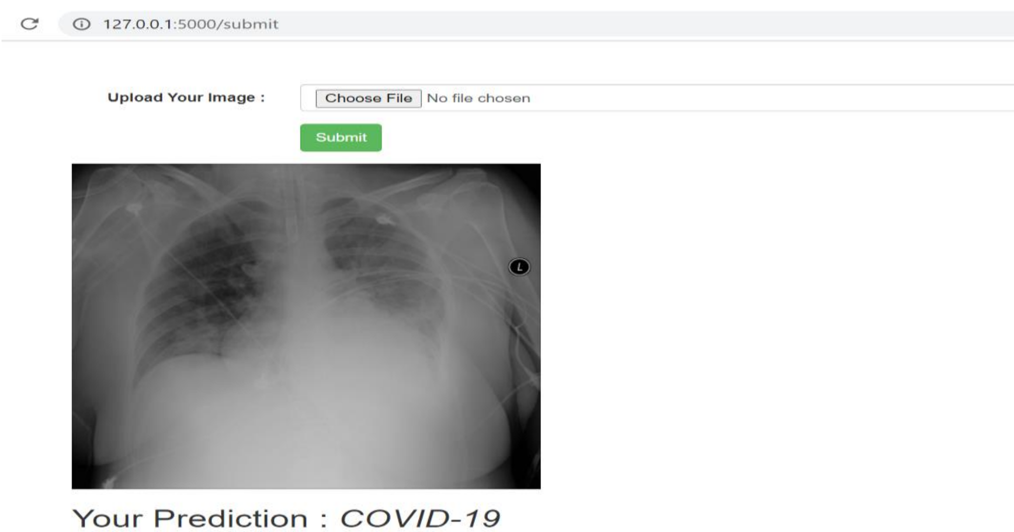
28. Define the Routes

```python
# routes
@app.route("/", methods=['GET', 'POST'])
def main():

    return render_template("final.html")

@app.route("/submit", methods = ['GET', 'POST'])
def get_output():
    if request.method == 'POST':
        img = request.files['my_image']

        img_path = "static/" + img.filename
        img.save(img_path)

        p = predict_label(img_path)

    return render_template("final.html", prediction = p, img_path = img_pat


if __name__ =='__main__':
```

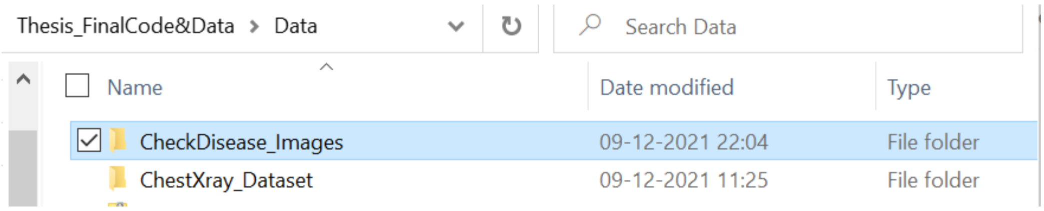
29. To implement it without any problem, store the app.py file and model.h5 in the same folder. Also create to more folders static and template. Static to store uploaded image and Templates to render the HTML file.



30. Webpage on local machine This is a webpage is designed to work on local machine only but I have also used Ngrok function and designed the same web page in Google colab to communicate from outside and access the web page from anywhere.



31. Test the Web-app using images in the folder **CheckDisease_Images**.



References

Adusumilli, G., 2020. Image Recognition using Pre Trained Xception Model in 5 steps. Analytics Vidhya. URL https://medium.com/analytics-vidhya/image-recognition-using-pre-trained-xception-model-in-5-steps-96ac858f4206 (accessed 12.15.21).

Ankit, U., 2021. Image Classification of PCBs and its Web Application (Flask) [WWW Document]. Medium. URL https://towardsdatascience.com/image-classification-of-pcbs-and-its-web-application-flask-c2b26039924a (accessed 12.15.21).

S, H., 2021. Activation Functions : Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep…. Medium. URL https://himanshuxd.medium.com/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e (accessed 12.15.21).

tf.keras.applications.xception.Xception | TensorFlow Core v2.7.0 [WWW Document], n.d. . TensorFlow. URL https://www.tensorflow.org/api_docs/python/tf/keras/applications/xception/Xception (accessed 12.15.21).