National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Shovan Roy
Student ID: X18201156

School of Computing
National College of Ireland

Supervisor: Mr. Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Shovan Roy

Student ID: X18201156

Programme: Data Analytics **Year:** 2019-2020

Module: MSc Research Project

Lecturer: MR. Hicham Rifai

Submission Due Date: 17/08/2020

Project Title: Configuration Manual

Word Count: 720 **Page Count:** 12

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: Shovan Roy

Date: 15/08/2020

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

Shovan Roy
X18201156

1 Introduction

This Configuration manual represents all the necessary requirements and details for the Project "COVID-19 Detection using Deep Learning". The 2nd section shows the Hardware requirements. The third section focuses on all the software required to be able to run this project. The 4th section Shows where to get the data from. The 5th section depicts the library module requirements of the project. The rest of the sections are divided into parts of workflow of the code, their execution, result and their evaluation.

2 Hardware Requirements

ASUS ROG 64 bits windows 10 operating system has been used for the project.

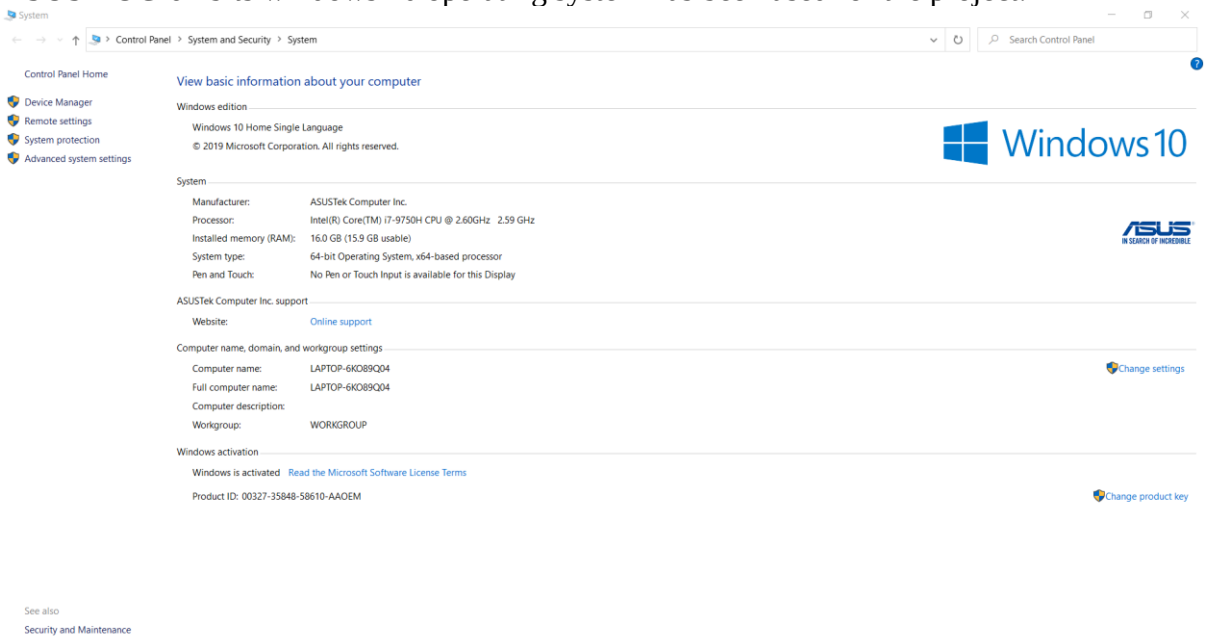

Figure 1 Hardware Requirements

3 Software Requirements

The Entire Script has been implemented using python 3.8[1]. The Anaconda Navigator[2][3] is required because the whole program has been done in Jupyter Notebook which in included in Anaconda Navigator.

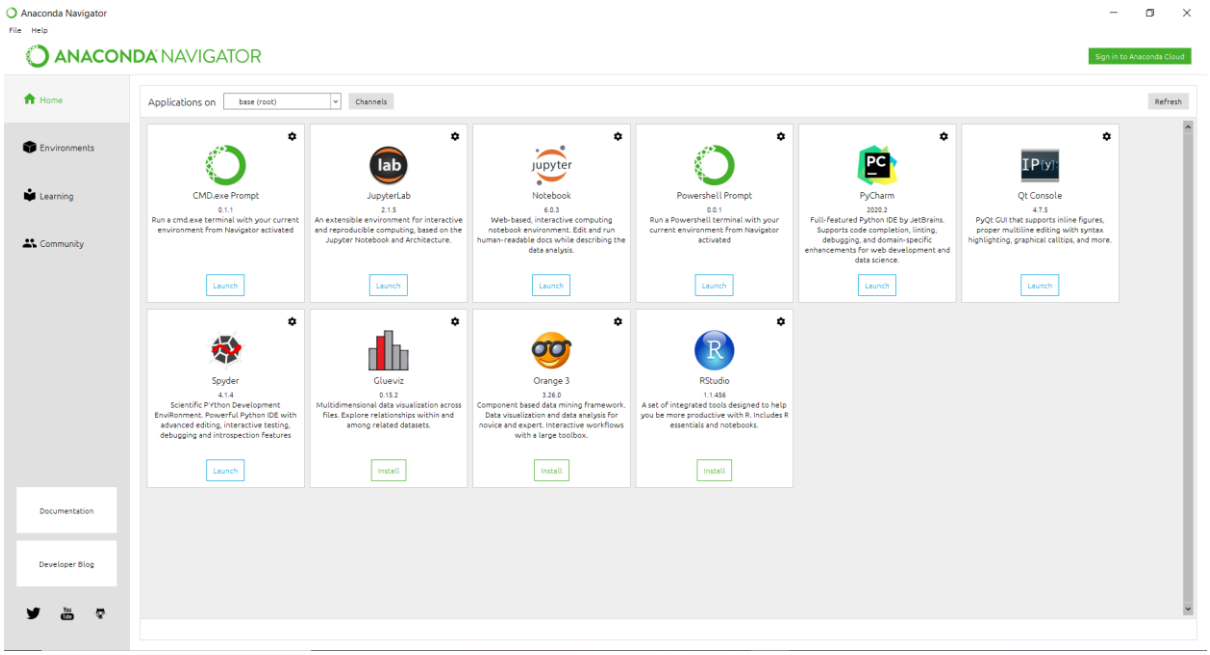

Figure 2: Anaconda Navigator

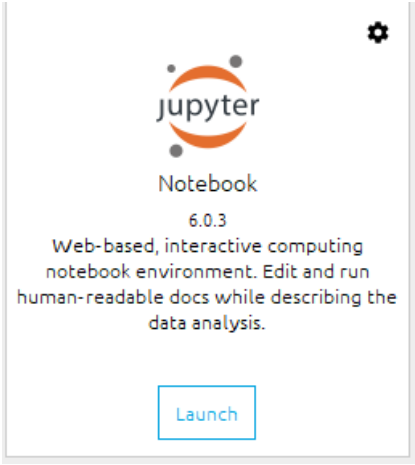

Figure 3: Jupyter Notebook

4 Data Requirements

The Data required for the project can be Downloaded from the Github[4] and contains no personal data and is in accordance with the GDPR Guidelines.

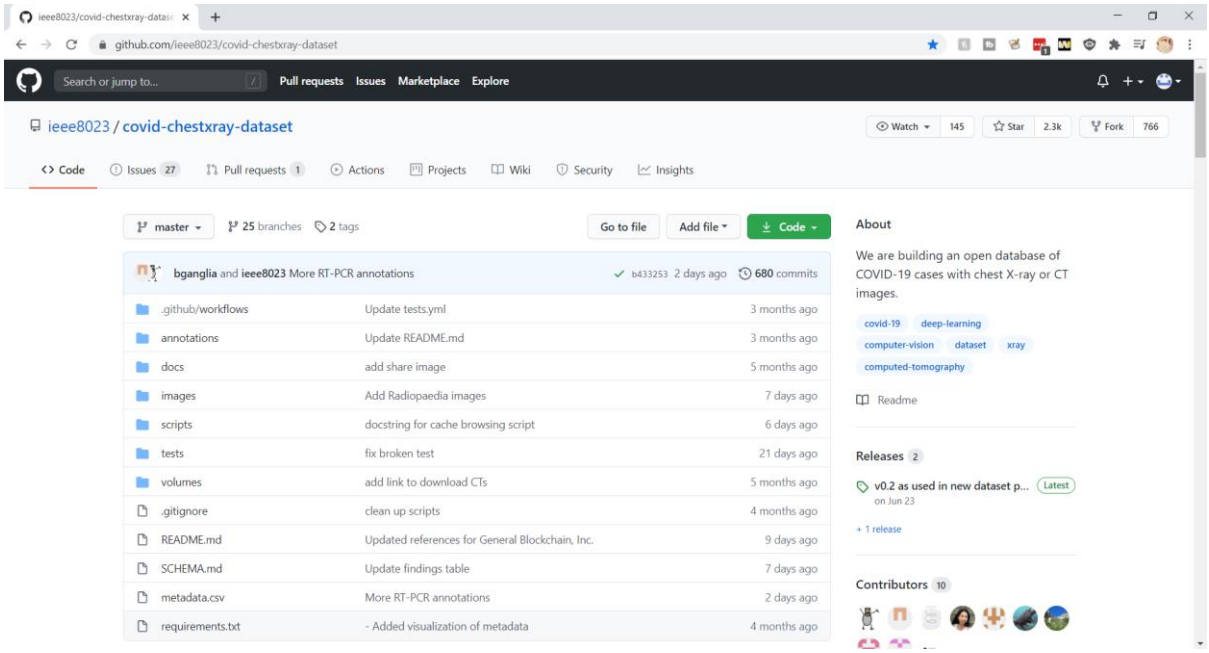

Figure 4: Github Page to be able to download Data

5 Installing Required Library Modules

These are the python libraries that need to be pip installed and imported for the project.

```
tensorflow==2.3.0
pandas==1.0.5
numpy==1.18.5
matplotlib==3.2.2
bokeh==2.1.1
```

Figure 5: Python Libraries & versions

6 Data Pre-processing

This is a part of pre-processing where the data is taken and has been divided into Test and Train sets.

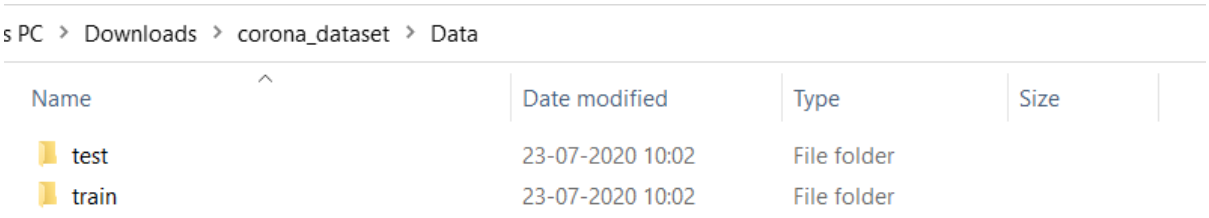

Figure 6: Data divided into Training and Testing set

The Data is further divided into folders of Covid chest X-rays and Normal chest X-rays.

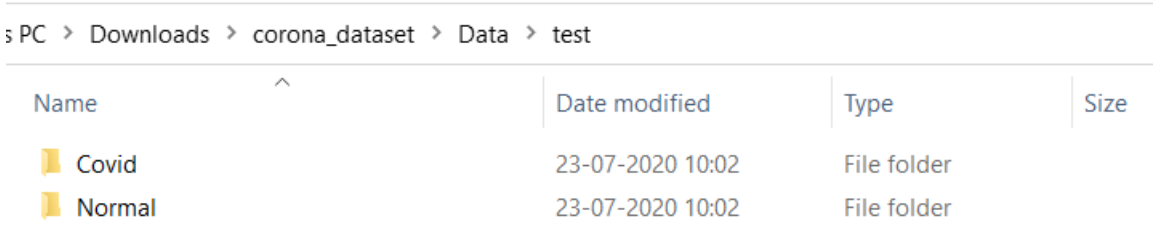

Figure 7: Data Divided into Covid & Normal

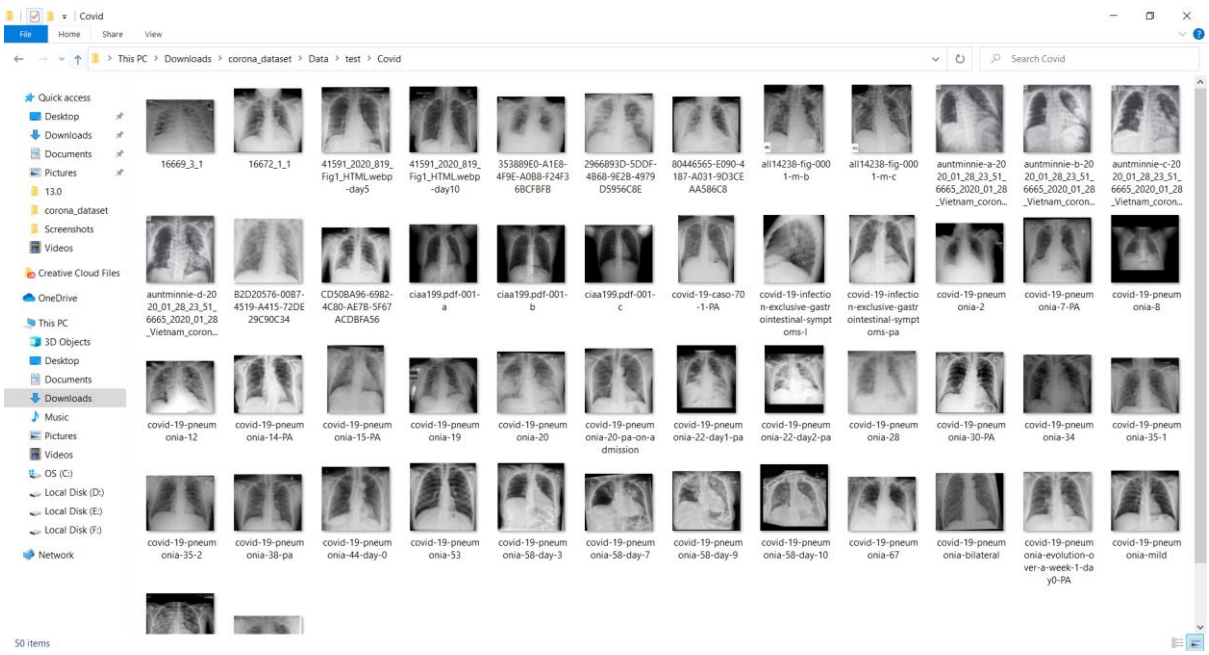

Figure 8: Covid Chest X-rays

7 Creating the Convolutional Neural Network (CNN)

Using tensorflow and Keras for CNN.

```
# Initialising the CNN
classifier = Sequential()

classifier.add(Conv2D(filters=32, kernel_size=3, padding="same", input_shape=(224, 224, 3),
                                            activation='relu'))

classifier.add(MaxPooling2D(pool_size=2, strides=2, padding='valid'))

classifier.add(Conv2D(filters=64, kernel_size=3, padding="same", activation="relu"))
classifier.add(MaxPooling2D(pool_size=2, strides=2, padding='valid'))


classifier.add(Conv2D(filters=64, kernel_size=3, padding="same", activation="relu"))
classifier.add(Dropout(0.2))
classifier.add(MaxPooling2D(pool_size=2, strides=2, padding='valid'))


classifier.add(Conv2D(filters=128, kernel_size=3, padding="same", activation="relu"))
classifier.add(Dropout(0.2))
classifier.add(MaxPooling2D(pool_size=2, strides=2, padding='valid'))

classifier.add(Conv2D(filters=256, kernel_size=3, padding="same", activation="relu"))
classifier.add(Dropout(0.2))
classifier.add(MaxPooling2D(pool_size=2, strides=2, padding='valid'))

classifier.add(Conv2D(filters=64, kernel_size=3, padding="same", activation="relu"))
classifier.add(MaxPooling2D(pool_size=2, strides=2, padding='valid'))


classifier.add(Flatten())

classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))

classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
classifier.summary()

```

Figure 9: CNN Model

8 Pre-processing and Loading Image set

2. Preprocessing & Loading the image set

In [27]:

```
#Fitting the CNN to the images

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1. / 255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

training_set = train_datagen.flow_from_directory('../corona dataset/Data/train',
                                                 target_size=(224,224),
                                                 batch_size=32,
                                                 class_mode='binary')

test_set = test_datagen.flow_from_directory('../corona dataset/Data/test',
                                            target_size=(224,224),
                                            batch_size=32,
                                            class_mode='binary')

```

Figure 10: Preprocessing and loading images

9 Training the Model

3. Training the Model

In [19]:

```
history = classifier.fit(training_set,
                         steps_per_epoch=4,
                         epochs=25,
                         validation_data=test_set,
                         validation_steps=4)

```

Figure 11: Training the CNN Model

10 Passing the Test set through the model

4. Evaluating the Test Set

In [20]:

```
# evaluation on test set
loaded_model = tf.keras.models.load_model('my_model.h5')
loaded_model.evaluate(test_set)

```

Figure 12: Putting the test set through the model

In [24]:

```
# for whole test set  {'Covid': 0, 'Normal': 1}
from keras.preprocessing import image
import numpy as np
import os
import tensorflow as tf

# image folder
folder_path = '../corona dataset/original test set'
# path to model
model_path = './my_model.h5'

# load the trained model
classifier = tf.keras.models.load_model('./my_model.h5')
classifier.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])

# load all images into a list
images = []
for img in os.listdir(folder_path):
    img = os.path.join(folder_path, img)
    img = image.load_img(img, target_size=(224,224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    images.append(img)

# stack up images list to pass for prediction
images = np.vstack(images)
classes = classifier.predict_classes(images, batch_size=10)
print(classes)

```

Figure 13: Classifying the test set

11 Plotting Accuracy and Loss

In [21]:

```
#plot accuracy and loss
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

Figure 14: Plotting Accuracy & Loss of the Model

12 Classifying single images

In [28]:

```
# for only one prediction
import numpy as np
from keras.preprocessing import image

test_image = image.load_img('../corona dataset/Data/test/Normal/IM-0283-0001.jpeg',target_size=(224,224
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis=0)
result = classifier.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'Normal'
else:
    prediction = 'Covid'
print(prediction)
```

```
Normal
```

In [29]:

```
# for only one prediction
import numpy as np
from keras.preprocessing import image

test_image = image.load_img('../corona dataset/Data/train/Covid/16654_1_1.png',target_size=(224,224))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis=0)
result = classifier.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'Normal'
else:
    prediction = 'Covid'
print(prediction)

```

```
Covid
```

Figure 15: Classifying Images

13 Plotting Confusion Matrix

```
In [25]:
# plot confusion matrix
y_pred = []
y_test = []
import os

for i in os.listdir("../corona dataset/Data/test/Normal"):
    img = image.load_img("../corona dataset/Data/test/Normal/" + i, target_size=(224,224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    p = classifier.predict_classes(img)
    y_test.append(p[0, 0])
    y_pred.append(1)

for i in os.listdir("../corona dataset/Data/test/Covid"):
    img = image.load_img("../corona dataset/Data/test/Covid/" + i, target_size=(224,224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    p = classifier.predict_classes(img)
    y_test.append(p[0, 0])
    y_pred.append(0)

y_pred = np.array(y_pred)
y_test = np.array(y_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_pred, y_test)
import seaborn as sns

sns.heatmap(cm, cmap="plasma", annot=True)
```

Figure 16: Plotting a Confusion Matrix

14 Evaluating Classification Report

The Classification Report[5] are necessary for Evaluation of any model.

```
In [26]:
from sklearn.metrics import classification_report

print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        50
           1       0.93      1.00      0.96        50

    accuracy                           0.96       100
   macro avg       0.96      0.96      0.96       100
weighted avg       0.96      0.96      0.96       100
```

Figure 17: Classification Report

References

Python Release Python 3.8.5 (no date) *Python.org*. Available at: https://www.python.org/downloads/release/python-385/ (Accessed: 15 August 2020).

Anaconda | Individual Edition (no date) *Anaconda.* Available at: https://www.anaconda.com/products/individual (Accessed: 15 August 2020).

Classification Report — Yellowbrick v1.1 documentation (no date). Available at: https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html (Accessed: 15 August 2020).

Cohen, J. P. (2020) *ieee8023/covid-chestxray-dataset*. Available at: https://github.com/ieee8023/covid-chestxray-dataset (Accessed: 15 August 2020).

Package repository for anaconda :: Anaconda Cloud (no date). Available at: https://anaconda.org/anaconda/repo (Accessed: 15 August 2020).