National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Pavan Doddaiah Kalpana
Student ID: 19201613

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Pavan Doddaiah Kalpana
Student ID:	19201613
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Christian Horn
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	282
Page Count:	7

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:	
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

Pavan Doddaiah Kalpana
19201613

1 Introduction

This document explains how to implement the research and configuration to set it up on any machines with any issues to have arisen.

2 System Configuration

2.1 Hardware Configuration

- Processor: Intel Core i5-6200U, Quad-Core Operating System: Windows 10, 64 Bit
- CPU: 8GB DDR4 RAM, 1600Mhz
- GPU: Nvidia GeForce 940M, 1GB
- Storage: 1TB HDD

2.2 Software Configuration

- Python 3.8.12
- Jupyter Notebook

2.3 Libraries Required

- Tensorflow 2.3.0
- Keras 2.3.0
- Numpy 1.21.2
- Seaborn 0.11.2
- Pandas 1.3.4
- Matplotlib 3.4.3
- Imutils 0.5.4
- Opencv_python 4.5.4.48
- Ipython 7.29.0
- Scikit_learn 1.0.1

3 Environmental Setup

To avoid incompatibility concerns, the study used several downgraded Python libraries. As a result, a new environment in Anaconda was created to meet all of the criteria. As an example, consider the following:

1. Open Anaconda Navigator. Click on 'Environments' tab. Click on 'Create' in the bottom.

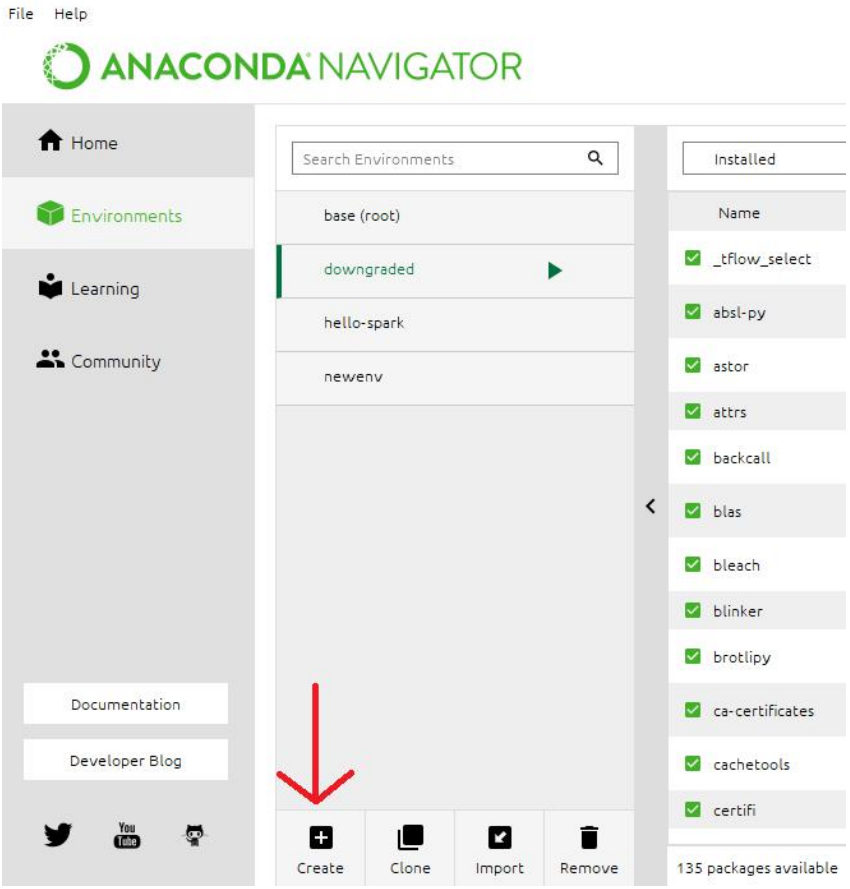

Figure 1: Anaconda Navigator

2. Open Anaconda Prompt. Type 'activate *your_env_name*.

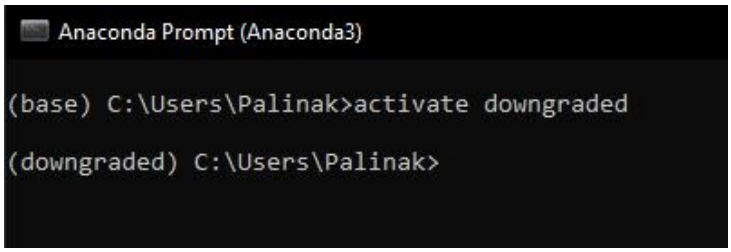

Figure 2: Anaconda Prompt

3. Use 'pip install' to install all of the essential libraries in this environment. Select

the environment generated before starting Jupyter notebook from Anaconda navigator.

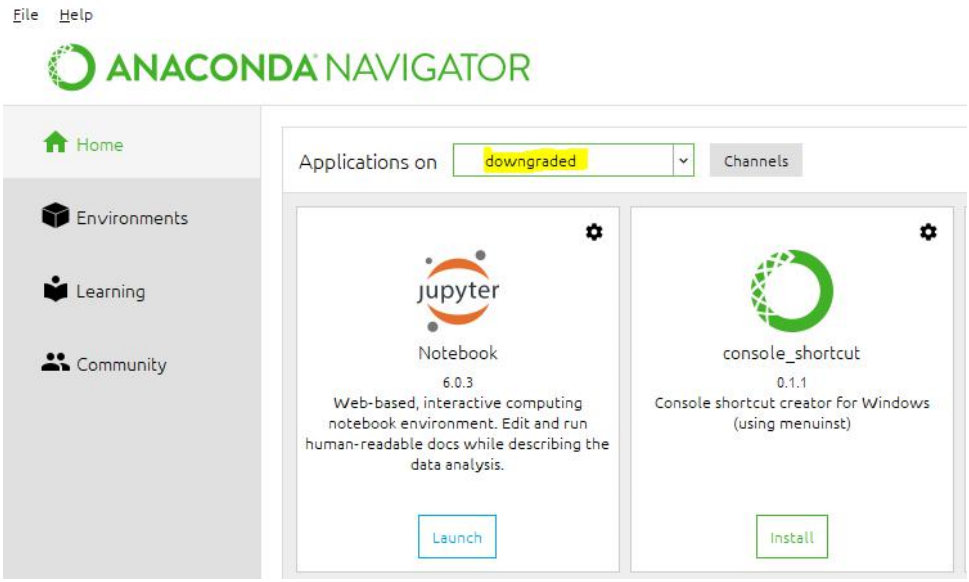

Figure 3: Anaconda Environment

4 Research code

```
In [1]:
# importing required libraries
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from imutils import paths
import numpy as np
import warnings
import shutil
import random
import os
import cv2
import seaborn as sns
from sklearn.metrics import confusion_matrix
from IPython.display import display
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras import utils as keras_utils
from tensorflow.keras import backend

from tensorflow.keras import layers, models

from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications import VGG19, VGG16
from tensorflow.keras.optimizers import Adam, RMSprop, SGD
from tensorflow.keras.layers import *
from tensorflow.keras import Model

from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.applications.mobilenet import preprocess_input
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint,
CSVLogger

import matplotlib
```

Figure 4: Importing the required libraries

```
# setting up basic values for graphs
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = [10, 5]
warnings.filterwarnings("ignore", category=FutureWarning)
```

```
# setting batch size and dataset path
batch_size = 32
dataset = 'COVID-Datasets'
imagePaths = sorted(list(paths.list_images(dataset)))
random.seed(42)
random.shuffle(imagePaths)
```

```
# initializing the array to store data and label
imagesData = []
labels = []
```

Figure 5: Initializing the variable for model and datasets

```
# converting string lable into numerical value to process
codeLabel = {1: 'COVID', 0: 'Normal', 2: "Viral_Pneumonia"}
labelCode = {}
for k, v in codeLabel.items():
    labelCode[v] = k
```

```
# creating model directory to store the trained models
if not os.path.isdir("./models"):
    print("Creating directory to store models")
    os.mkdir("./models")

temp = {}
np.save('./models/labelName.npy', codeLabel)
```

```
Creating directory to store models
```

Figure 6: Creating the directory to store models

```
# this block is to load the images and resize the image
for i, imagePath in enumerate(imagePaths):
    try:
        image = cv2.imread(imagePath, 1)
        image = image[..., ::-1]
        image = cv2.resize(image, (224, 224))
        image = (image / 255.).astype(np.float32)

        label = imagePath.split(os.path.sep)[-2]
        if label not in temp:
            temp[label] = 1
        else:
            temp[label] += 1
        if temp[label] <= 3600:
            imagesData.append(image)
            labels.append(labelCode[label])

    except (FileNotFoundError, Exception) as e:
        print(imagePath)
        pass
```

Figure 7: Load and Resize the images

```
# converting list into array
labels = np.array(labels)
print("Output data shape : ", labels.shape)

# splitting dataset
trainX, testX, trainY, testY = train_test_split(imagesData, labels, test_size=0.2)

print(np.array(trainX).shape)
print(np.array(trainY).shape)
print(np.array(testX).shape)
print(np.array(testY).shape)
```

```
Output data shape :  (8545,)
(6836, 224, 224, 3)
(6836,)
(1709, 224, 224, 3)
(1709,)
```

Figure 8: Splitting dataset for training and testing

```python
# Create the directory structure
base_dir = 'dataDir'
if not os.path.isdir(base_dir):
    print("Creating base_dir")
    os.mkdir(base_dir)

# train_dir
train_dir = os.path.join(base_dir, 'train_dir')
if not os.path.isdir(train_dir):
    print("Creating train_dir")
    os.mkdir(train_dir)

# val_dir
val_dir = os.path.join(base_dir, 'val_dir')
if not os.path.isdir(val_dir):
    print("Creating val_dir")
    os.mkdir(val_dir)
```

```
Creating base_dir
Creating train_dir
Creating val_dir
```

Figure 9: Creating data dir for generator

```python
# storing training images into separate directory
idx = 1
for image, label in zip(trainX, trainY):
    label_dir = os.path.join(train_dir, str(label))
    if not os.path.isdir(label_dir):
        os.mkdir(label_dir)
    dst = os.path.join(label_dir, "label_{}.png".format(idx))
    image = cv2.convertScaleAbs(image, alpha=(255.0))
    cv2.imwrite(dst, image)
    idx += 1
```

```python
# storing validation images into separate directory
idx = 1
for image, label in zip(testX, testY):
    label_dir = os.path.join(val_dir, str(label))
    if not os.path.isdir(label_dir):
        os.mkdir(label_dir)
    dst = os.path.join(label_dir, "label_{}.png".format(idx))
    image = cv2.convertScaleAbs(image, alpha=(255.0))
    cv2.imwrite(dst, image)
    idx += 1
```

```python
# Set Up the Generators
train_path = 'dataDir/train_dir'
val_path = 'dataDir/val_dir'

datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_gen = datagen.flow_from_directory(train_path, target_size=(224, 224), batch_size=32)

val_gen = datagen.flow_from_directory(val_path, target_size=(224, 224), batch_size=32)

test_gen = datagen.flow_from_directory(val_path, target_size=(224, 224), batch_size=1, shuffle=False)
```

```
Found 6836 images belonging to 3 classes.
Found 1709 images belonging to 3 classes.
Found 1709 images belonging to 3 classes.
```

Figure 10: Storing the processed images and data generator

```
print("Deleting unwanted folder")
shutil.rmtree("./dataDir")
```

```
Deleting unwanted folder
```

Figure 11: Deleting the directory