Configuration Manual

MSc Research Project
Data Analytics

Melwin Francis Rodrigues
Student ID: X18183000

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	MELWIN FRANCIS RODRIGUES		
Student ID:	X18183000		
Programme:	Data Analytics	**Year:**	2020
Module:	MSc Research Project		
Lecturer:	Dr. Catherine Mulwa		
Submission Due Date:	17/08/2020		
Project Title:	Breast Cancer Detection Using Deep Learning Techniques		
Page Count	12		
Word Count:	1062		

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: MELWIN FRANCIS RODRIGUES

Date: 17/08/2020

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

Melwin Francis Rodrigues
Student ID: X18183000

1 Introduction

The configuration manual is a step by step guide for project guidance in development, installation, implementation and deployment of the project 'Breast Cancer Detection using Deep learning techniques' presented in technical report. The agenda of this report is to assist and take through each step to gain the desired output and results which are presented in technical report. The entire project is implemented by using multiple technologies, libraries, hardware and software configurations.

1.1 Project Overview

The aim of this project is to classify and detect the breast cancer in the preliminary stage before it gets dangerous for the women life. The methods used are Dense Net 121, CNN and Inception V3 for image classification. The dense Net 121 gave better results, and this could help in better understanding and detection of cancer in women in histopathological images.

2 Pre-requisites

The pre-requisites are as follows. The software and hardware configurations are given below. The GPU (Graphics Processing Unit) is a must to train the model for such a large images dataset.

2.1 Hardware requirements

- Processor Required: AMD Ryzen 5 3550H
- RAM: 8GB
- GPU: Nvidia Tesla K80 (Google Colab)
- ROM: Minimum 10GB
- Operating System: Windows 10

2.2 Software requirements

- Programming Language: Python.
- Development Tools: Jupyter Notebook, Google Colab, Microsoft Excel.

3 Software Installation Guide

3.1 Anaconda Navigator and Jupyter Notebook

- Download Anaconda installer.
- Double click on installer to start.
- Check and address the Read Me and License agreement.
- Install it by clicking install button “Just Me’ unless if installing for other users.
- Select a destination directory or any of your preferred directory.
  Follow the installation guiding for graphical assistance Link

4 Project Implementation Guide

This section talks about implementation of the project. All the codes, packages and logic are explained in brief in this section.

4.1 Data Understanding and Pre-processing

For any machine learning project, the understanding of the data is very important. Below Figure1 show the snapshot of the code. The original dataset has multiple sub folders. Fold1 has Test and train folders. Then each Test and train individually have 40x, 100x, 200x, 400x folders which again have sub folders named B for benign and M for malign. First the file named Thesis.ipynb is to be run the snapshot is given below.

```
In [5]: import shutil
        import os


        source = 'C:/Melwin/COLLEGE PORTION/sem 3 thesis/breast-cancer-dataset-from-breakhis final dataset/Final dataset/fold1'
        dest1 = 'C:/Melwin/COLLEGE PORTION/sem 3 thesis/breast-cancer-dataset-from-breakhis final dataset/Final dataset/melwin'


        files = os.listdir(source)
        #files.pop(0)
        print(files)
        for i in files:
            sub_folder = os.listdir(source+"/"+i)
        print(sub_folder)
        dict_1 = {}
        for i in sub_folder:
            #print(i)

            list2 = os.listdir(source+"/"+files[0]+"/"+i)
            if ".DS_Store" in list2:
                list2.pop(0)
            dict_1[i] = list2
        print(dict_1)

        ['test', 'train']
        ['100X', '200X', '400X', '40X']
        {'100X': ['B_100X', 'M_100X'], '200X': ['B_200X', 'M_200X'], '400X': ['B_400X', 'M_400X'], '40X': ['B_40X', 'M_40X']}
```

Figure 1: Data Source folders and subfolders

The below Figure2 is used for moving the image into 2 separate folders called M (malign) and B (Benign).

```
dir1= source+"/"+files[0]
print(dir1)
for key,value in dict_1.items():
    for v in value:
        list1= os.listdir(dir1+"/"+key+"/"+v)
        for e in list1:
            if v[0]=="M":
                shutil.move(dir1+"/"+key+"/"+v+"/"+e, dest1+"/"+"M")
            else:
                shutil.move(dir1+"/"+key+"/"+v+"/"+e, dest1+"/"+"B")
```

Figure 2: Moving the data into separate folders.

The metadata is created based on the dataset as from the above all the images are categorized into 2 categories name malign and benign which are moved in 2 separate folders called M and B shown in Figure3.

The below code is used for creating metadata from the image dataset for benign image (B). CSV file is created (benign.csv) and separate folder named 'B'

```
import os
import pandas as pd

list_B = os.listdir('C:/Melwin/COLLEGE PORTION/sem 3 thesis/breast-cancer-dataset-from-breakhis final dataset/Final dataset/melw

df_B = pd.DataFrame (list_B, columns=['Benign'])
df_B.to_csv(r'C:/Melwin/COLLEGE PORTION/sem 3 thesis/breast-cancer-dataset-from-breakhis final dataset/Final dataset/melwin/beni
```

The below code is used for creating metadata from the image dataset for Malign image (M). CSV file is created (Malign.csv) and separate folder named 'M'

```
list_M = os.listdir('C:/Melwin/COLLEGE PORTION/sem 3 thesis/breast-cancer-dataset-from-breakhis final dataset/Final dataset/melw


df_M = pd.DataFrame (list_M, columns=['Malign'])
df_M.to_csv(r'C:/Melwin/COLLEGE PORTION/sem 3 thesis/breast-cancer-dataset-from-breakhis final dataset/Final dataset/melwin/Mali
```

Figure 3: Create CSV files(metadata)

For any dataset there should not be any data leakage as it would be a major concern when getting results. The below Figure4 is showing the code for checking the data leakage.

```
def check_leakage(benigncsv_df, Maligncsv_df, patient ):

    #Return True if there any patients are in both df1 and df2.


    benign_unique = benigncsv_df[patient].unique()
    Malign_unique = Maligncsv_df[patient].unique()

    Maligbenig_in_both_groups = set(benign_unique).intersection(set(Malign_unique))

    # leakage contains true if there is patient overlap, otherwise false.
    if len(Maligbenig_in_both_groups) >0:

        leakage = True # boolean (true if there is at least 1 patient in both groups)
    else:
        leakage = False

    return leakage
```

Figure 4: To check data Leakage

The below Figure5 is used for splitting of dataset into train and test. 60% of the data is split into train and 40% into test and validation equally.

```
X_train, X_test  = train_test_split(X, test_size=0.4, random_state=42, shuffle=True)
print("\nX_train:\n")
print(X_train.head())
print(X_train.shape)

print("\nX_test:\n")
print(X_test.head())
print(X_test.shape)
```

Figure 5: To split the dataset into 60, 40 ratio

The data was then pre-processed by using rescale, shear range, zoom and horizontal flip by using Image data generator.

```
from 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)
```

Figure 6: The reprocessing using rescale, horizontal flip

4.2 Implementation of models

Total of 3 machine learning models were used CNN, Dense Net 121 and Inception V3.

4.2.1 CNN

Below Figure7 is the code for CNN. The optimizer used is Adam. The activation function used are relu and sigmoid. The loss function is binary crossentropy because there are only 2 classes in the dataset. The file to run is CNN.ipynb this was implemented on google colab and performed by using GPU.

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras import layers
from keras.utils import to_categorical
import numpy as np
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import Adam, SGD
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

```
model = Sequential()
model.add(Conv2D(32, (5, 5), activation='relu', input_shape=(250,250,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(250, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(loss='binary_crossentropy',
              optimizer = Adam(lr=0.0001),
              metrics=['accuracy'])
```

Figure 7: Building the CNN model

The below Figure8 code is used for model accuracy which was 87.74%.

```
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()
```

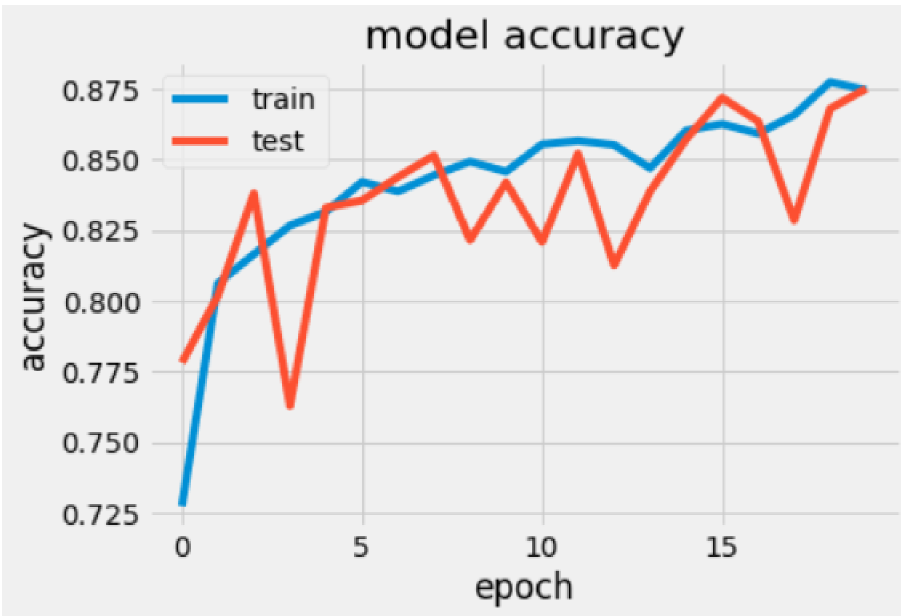

Figure 8: The model performance of train and test.

4.2.2 Inception V3

The below Fig9 is of inception V3 model with the starting layer set to false. The optimizer used is adam. The activation function is sigmoid, and the loss function is binary cross entropy. The File to run is Inception Final.ipynb which was implemented on google colab with GPU.

```
from tensorflow.keras.applications.inception_v3 import InceptionV3

pre_trained_model = InceptionV3(input_shape= (250,250,3), include_top=False, weights='imagen

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_
87916544/87910968 [==============================] - 1s 0us/step

for layer in pre_trained_model.layers:
  layer.trainable=False

from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import Adam, SGD
from keras.layers.normalization import BatchNormalization
from tensorflow.keras import layers
from tensorflow.keras import Model
x = pre_trained_model.output

x = layers.Flatten()(x)
x = layers.Dense(1024,activation = 'sigmoid')(x)
x = layers.BatchNormalization()(x)
x = layers.Dropout(0.4)(x)
x = layers.BatchNormalization()(x)
x = layers.Dense(1, activation='sigmoid')(x)
model = Model(pre_trained_model.input,x)
model.compile(optimizer = Adam(lr=0.0001), loss = 'binary_crossentropy', metrics = ['acc'])
```

Figure 9: Inception V3

To check the accuracy and loss below Figure10 is used the accuracy was 89.79%.

```
import numpy as np
N=85
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), history.history["acc"], label="train_acc")
plt.plot(np.arange(0, N), history.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on Dataset")
plt.xlabel("Epoch")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.show()
```

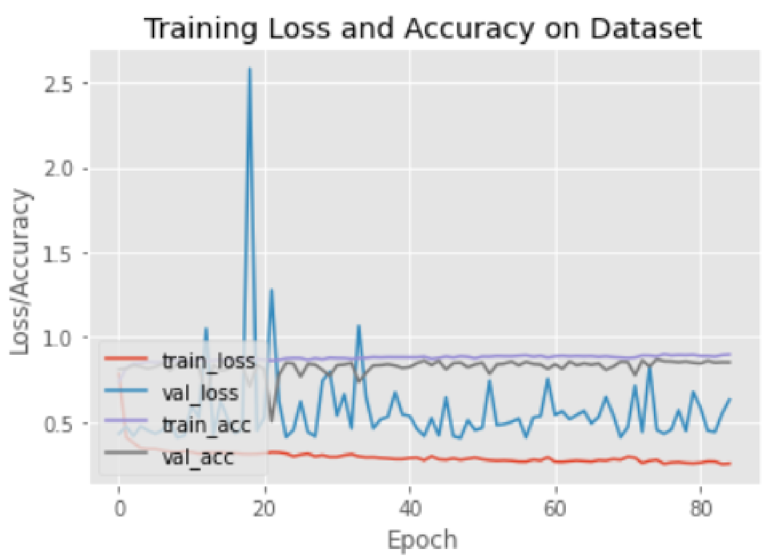

Figure 10

4.2.3 DenseNet121

The file to execute is DenseNet121.ipynb. The below Figure11 is used to build the model. The optimizer used is Adam. The loss function is binary crossentropy and activation function used is relu and sigmoid.

```
from keras.applications.densenet import DenseNet121
dn121 = DenseNet121(
    weights='imagenet',
    include_top=False,
    input_shape=(250, 250, 3))
```

```
def build_model(base, layer_units, num_classes):
    for layer in base.layers:
        layer.trainable = False

    x = base.output
    x = Flatten()(x)
    for num_units in layer_units:
        x = Dense(num_units, activation='relu')(x)
    predictions = Dense(num_classes, activation='sigmoid')(x)
    model = Model(inputs=base.input, outputs=predictions)
    return model
```

```
from keras.optimizers import Adam, SGD
from keras.layers import Activation, Dense, Flatten
from keras.models import Model, load_model

adam = Adam(lr=0.0001)
model = build_model(dn121, [1024], 1)
model.compile(adam, loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 11: Dense Net121

The below Figure12 shows the accuracy of test and train data with the number of epochs.

```
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()
```

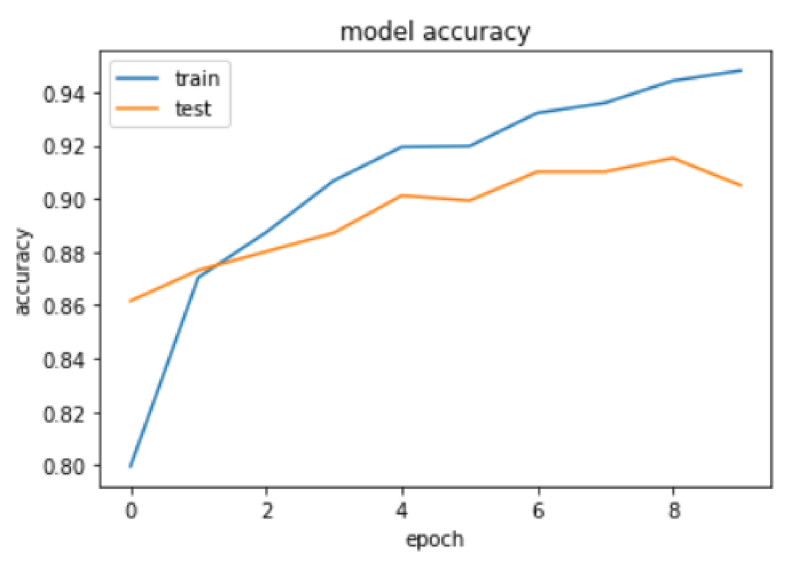

Figure 12: Model Accuracy