Configuration Manual

MSc Research Project
Data Analytics

Nivedita Vishwanath Hiremath
Student ID: x21108471

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nivedita Vishwanath Hiremath
Student ID:	x21108471
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr. Christian Horn
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	584
Page Count:	6

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:	17th September 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

Nivedita Vishwanath Hiremath
x21108471

1 Introduction

The purpose of this guide is to describe the implementation configuration and set up followed in this research experiment. This documentation gives complete information of software and hardware configuration and libraries used in this project. It also describes the coding process and the technique to be followed in order to run the code.

2 Local Machine System Configuration

The Figure 1 shows system configuration used in this project

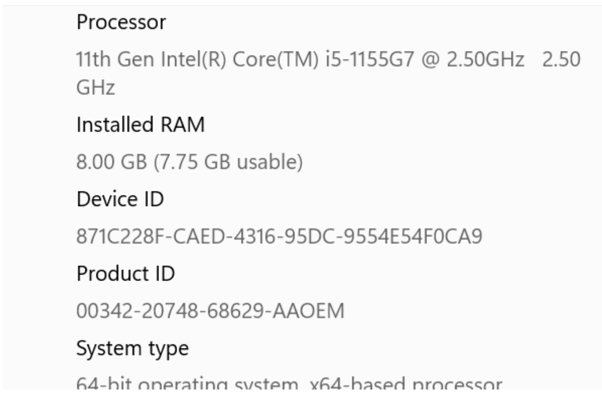

Figure 1: System Configuration

3 Dataset Collection

The dataset used in this project is BreakHis[1] collected from kaggle website. It consists of 7909 images with 4 different magnification levels namely 40X,100X,200X and 400X

4 JupyterLab SetUp

JupyterLab version of 3.3.2 was used to execute the program the and Figure 2 shows the configuarations of JupyterLab and python version of 3.9.7 was used throughout the work

[1]https://www.kaggle.com/datasets/ambarish/breakhis

```
Microsoft Windows [Version 10.0.22000.856]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nived>jupyter --version
Selected Jupyter core packages...
IPython          : 8.1.1
ipykernel        : 6.9.2
ipywidgets       : not installed
jupyter_client   : 7.1.2
jupyter_core     : 4.9.2
jupyter_server   : 1.15.6
jupyterlab       : 3.3.2
nbclient         : 0.5.13
nbconvert        : 6.4.4
nbformat         : 5.2.0
notebook         : 6.4.10
qtconsole        : not installed
traitlets        : 5.1.1
```

Figure 2: JupyterLab Environment

5 Conversion of RGB to HSV

The BreakHis dataset downloaded from the Kaggle was saved in folder named dataset_project in present working directory and the output of the converted HSV(Hue Saturation Value) images were updated in test folder in existing working directory.The libraries required are as shown in Figure 3

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from skimage.morphology import extrema
from skimage.morphology import watershed as skwater
import glob
import os
```

Figure 3: JupyterLab Environment

The steps followed in conversion of RGB to HSV are as follows-

- The libraries required to execute HSV are numpy,cv2,glob,matplotlib,os and skimage
- Test folder was created in the present working directory
- All the images of benign were loaded first from folder dataset_project using glob Figure 4

```
files = glob.glob('.\\dataset_project\\breast\\benign\\SOB\\**\\**\\**\\*.png',recursive = True)
parent_path = '.\\test\\benign\\SOB'
```

Figure 4: Load benign images

- The RGB images of benign are converted to HSV by giving purplish blue minimum threshold and maximum threshold range it is shown in Figure 5 and saved to folder test

```
cell_hsvmin  = (80,20,20)  #Lower end of the HSV range defining the nuclei
cell_hsvmax  = (150,190,190) #Upper end of the HSV range defining the nuclei
hsv          = cv2.cvtColor(filename_1,cv2.COLOR_BGR2HSV)
color_thresh = cv2.inRange(hsv, cell_hsvmin, cell_hsvmax)
masked = cv2.bitwise_and(filename_1,filename_1, mask=color_thresh)
cv2.imwrite(os.path.join(path_2 , r[-1]), masked)
```

Figure 5: HSV range

- All the images of malignant were loaded first from folder dataset_project using glob Figure 6

```
files = glob.glob('.\\dataset_project\\breast\\malignant\\SOB\\**\\**\\**\\*.png',recursive = True)
parent_path = '.\\test\\malignant\\SOB'
```

Figure 6: Load malignant images

- The RGB images of malignant are converted to HSV by giving purplish blue minimum threshold and maximum threshold range it is shown in Figure 5 and saved to folder test

6 Applying EfficientNet_B0 to HSV converted images and to EfficientNet_B0 without HSV

Steps followed in applying EfficientNet_B0-

- At different magnification levels the folders for HSV converted images and non converted images were segregated and saved into different folders
- Efficient_B0 was applied to HSV converted images and to without converted images at different magnification levels
- For all the images loaded at different magnification levels are differentiated into train, test and valid dataset Figure 7

```
# split training and validation set
valid_df = train_df.sample(frac=0.2)
train_df = train_df.drop(valid_df.index).reset_index(drop=True)
valid_df = valid_df.reset_index(drop=True)

test_df['set'] = 'test'
train_df['set'] = 'train'
valid_df['set'] = 'valid'
data_new = pd.concat([train_df,valid_df, test_df])
print(data_new)

# ax = sns.displot(data=data_new, x='label', col='set')

print('Training set')
print(train_df.label.value_counts())

print('\nValidation set')
print(valid_df.label.value_counts())

print('\nTest set')
print(test_df.label.value_counts())
```

Figure 7: Train Test Valid

- Data imbalance was handled on train data by upsampling benign images.

- The feature vectors of EfficientNet_B0 trained from ImageNet were loaded[2]
- All the images were further resized in resize_rescale() Figure 8 method as EfficientNet_B0 takes only images of (224,224,3) resolution.

```
def resize_rescale(image, label):
    img = tf.cast(image, tf.float32)
    img = tf.image.resize(img, [IMAGE_SIZE, IMAGE_SIZE])/255
    return img, label
```

Figure 8: Resize Rescale to EfficientNet_B0 image size

- The required tensorflow libraries for executing EfficientNet_B0 and matplotlib, seaborn library for visualisation imported for executing EfficientNet_B0 as shown in Figure 9

```
import os
import numpy as np
import pandas as pd

import tensorflow as tf
import tensorflow_hub as hub
from tensorflow.keras import layers
from tensorflow.keras.models import Model
import tensorflow_addons as tfa

from sklearn.metrics import *
import scikitplot as skplt

from functools import partial
import albumentations as A
import matplotlib.pyplot as plt
import seaborn as sns

AUTOTUNE = tf.data.experimental.AUTOTUNE
```

Figure 9: EfficientNet Libraries

- The model parameters are as set as below -
  - batch size, epochs values as Figure 10 were set and initial_learning_rate , maximal_learning_rate as shown in Figure 11

```
model_name = 'efficientnet_b0'
model_handle = model_handle_map.get(model_name)
IMAGE_SIZE = model_image_size_map.get(model_name, 224)
BATCH_SIZE = 64
EPOCHS = 12
```

Figure 10: Input Batch size and Epochs

  - optimiser and loss function was set in model.compile Figure 12

[2]https://tfhub.dev/tensorflow/efficientnet/b0/feature-vector/1

```
clr_scheduler = tfa.optimizers.CyclicalLearningRate(
    initial_learning_rate=2e-05,  maximal_learning_rate=7e-3,
    step_size=3*(SAMPLE_SIZE//BATCH_SIZE),
    scale_fn=lambda x: 1 / (2.0 ** (x - 1)),
    scale_mode='cycle'
```

Figure 11: Model Parameters

```
model.compile(
    optimizer=tf.keras.optimizers.SGD(learning_rate=clr_scheduler) ,
    loss=tf.keras.losses.BinaryCrossentropy(),
    metrics=METRICS
)
```

Figure 12: Optimizer Loss

- The sequential model building steps were followed as shown in Figure 13.
- All the evaluation were captured in training history method Figure 14.

```
    model = tf.keras.Sequential([
        layers.InputLayer(input_shape=(image_size, image_size, 3)),
        hub.KerasLayer(model_handle, trainable=True, name='base_model'),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(128, activation='relu'),
        layers.BatchNormalization(),
        layers.Dense(1, activation='sigmoid', name='classifier')
    ],name=model_name)
    model.build((None, image_size, image_size, 3))
    model.summary()
```

Figure 13: Model Build

```python
def training_history(history):
    accuracy = history['accuracy']
    val_accuracy = history['val_accuracy']

    loss = history['loss']
    val_loss = history['val_loss']

    epochs_range = range(len(history['loss']))

    plt.figure(figsize=(16, 4))
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, accuracy, label='Training accuracy')
    plt.plot(epochs_range, val_accuracy, label='Validation accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Loss')

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')

    plt.show()
    return None
```

Figure 14: Evaluation Metrics Capture