Configuration Manual

MSc Research Project
Data Analyticcs

Prasad Balasaheb Thorat
Student ID: 18185711

School of Computing
National College of Ireland

Supervisor: Dr Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Prasad Balasaheb Thorat
Student ID:	x18185711
Programme:	Data Analyticcs
Year:	2018
Module:	MSc Research Project
Supervisor:	Dr Christian Horn
Submission Due Date:	28/09/2020
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	9

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:	28th September 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

Prasad Balasaheb Thorat
x18185711

1 Hardware/Software Requirements

The configuration manual describes the steps required while running the scripts implemented for the research project. This manual will help to run the code without any problems. This manual also includes the information about hardware configuration of the system in which code were executed. The minimum required configuration for the system is also mentioned.

2 System Specification

2.1 Hardware Requirements

The hardware specifications of the system on which the research project is implemented are as follows.
Processor: Intel Core i5 – 8265U CPU @ 1.60GHz 1.80GHz
RAM: 8 GB
Storage: 128GB SSD/1TB HDD
Operating System: 64-bit operating system, Windows 10 Home

2.2 Software Requirements

This research project used following programming tools.
1.Google Colaboratory (Cloud based Jupyter notebook environment),
2.Python version 3,
3.Microsoft Excel
4.Overleaf

3 Enviroment Setup

3.1 Google Colaboratory

This section will help to setup Google Colaboratory enviroment. The following screenshots are included for better understanding.

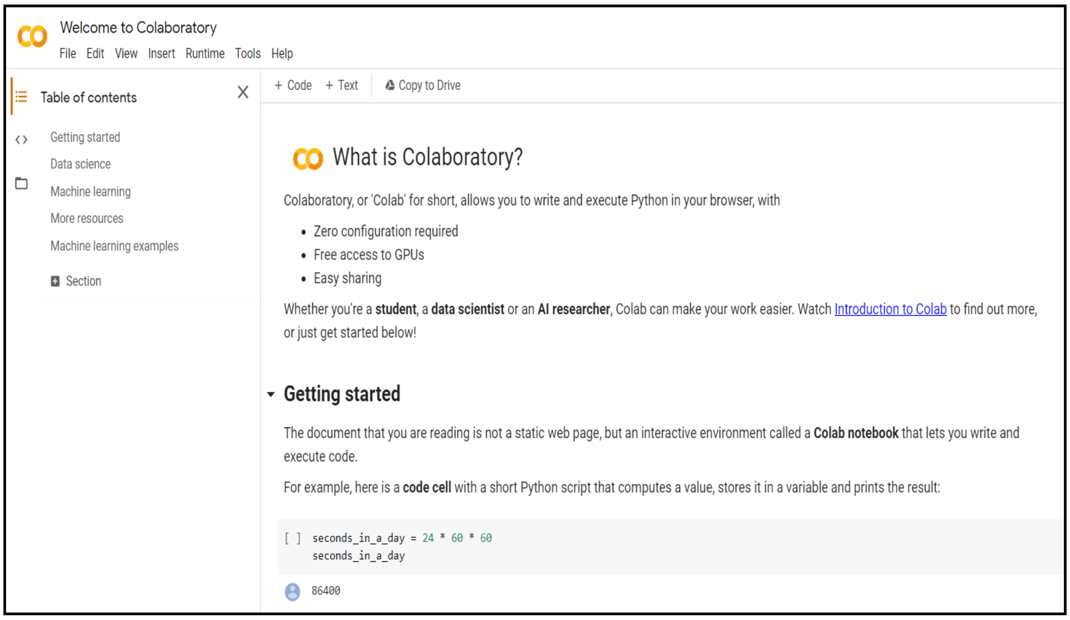

Figure 1: **Google Colaboratory Setup**

4 Data Source

1. This research project used the dataset of histopathological images which are publicly available as shown in Figure 2.

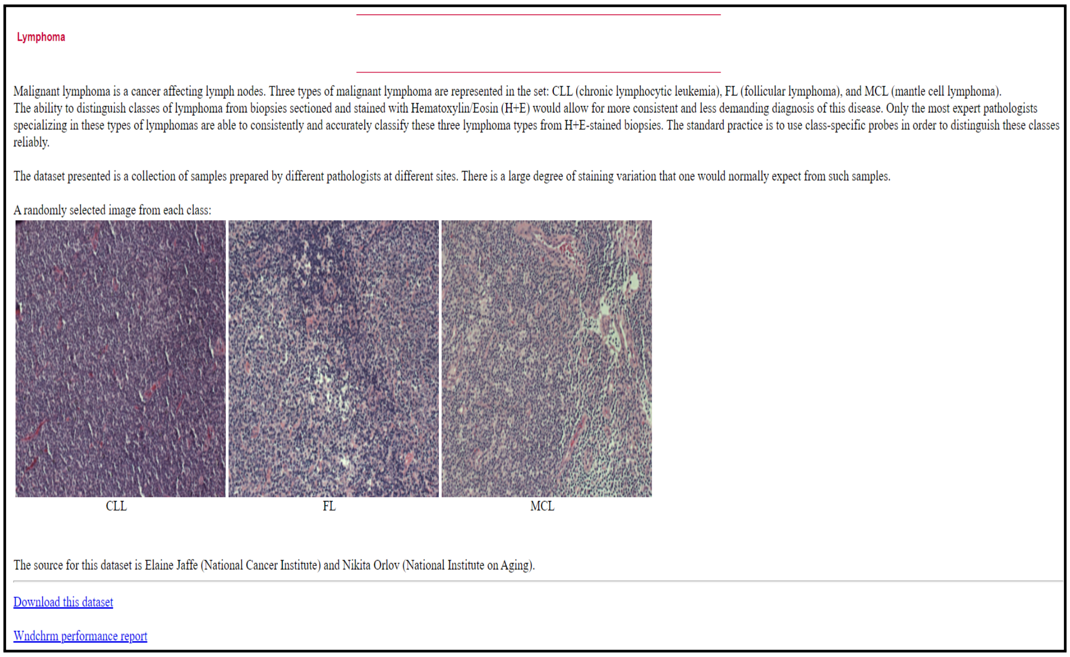

Figure 2: **Data Source**

2. Upload the downloaded dataset to google drive from gmail account. After that mount the google drive to colab notebook as shown in Figure 3. Click on the url and select the gmail account and enter the authentication code.

Figure 3: **Mounting Drive**

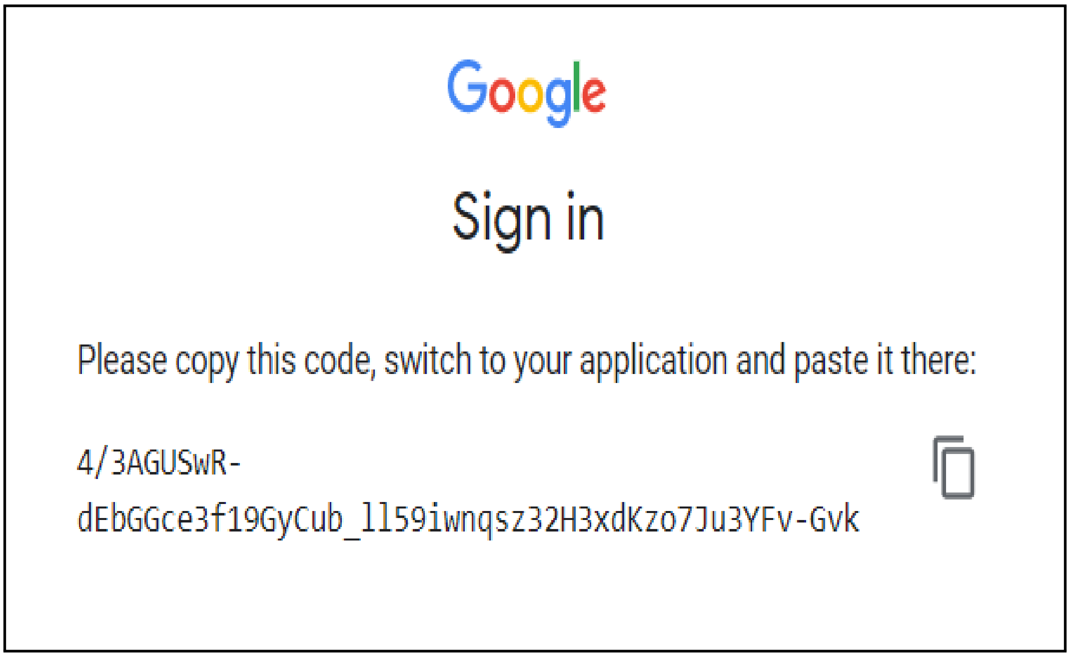

Figure 4: **Authentication Code**

5 Implementation

Following are the necessary libraries that are required to build an image classification model.

1. TensorFlow
2. NumPy
3. Matplotlib
4. opencv python
5. pandas
6. keras-preprocessing
7. Python Imaging Library(PIL)
8. keras.applications
9. Sklearn

5.1 Data Preprocessing

As a part of preprocessing, histogram normalization and image augmentation was performed on entire dataset. The following Figure 5 shows the required libraries.

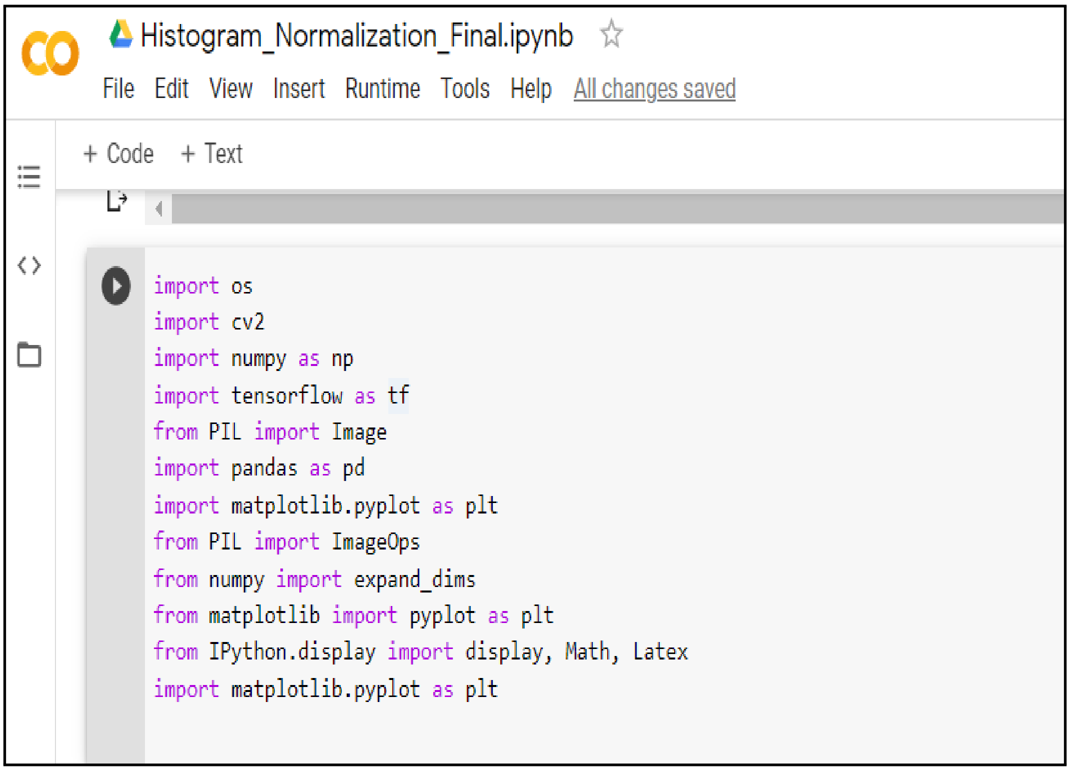

Figure 5: **libraries for Augmentation and Histogram Normalization**

The Figure 6 shows the function written for Data Augmentation. The function will take path of all the images that needs to be augmented and will write the augmented images in respective folder. The augmentation methods available in tensorflow are applied. Similarly The Figure 7 demonstrates the code for histogram normalization. A function

Data_Augmentation.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
from PIL import Image
import cv2

dest = "/content/gdrive/My Drive/Dataset/Augmentation/FL/"

def augmentation(path):

  for image in os.listdir(path):

    img_path = os.path.join(path + image)
    img_load = cv2.imread(img_path)

    flippedrl = tf.image.flip_left_right(img_load)
    fliprl = np.asarray(flippedrl)
    cv2.imwrite(dest + image[:-4] + '_fliprl' + '.tif', fliprl)

    rotated = tf.image.rot90(img_load)
    rot = np.asarray(rotated)
    cv2.imwrite(dest + image[:-4] + '_rotate90' + '.tif', rot)

    flippedud = tf.image.flip_up_down(img_load)
    flipud = np.asarray(flippedud)
    cv2.imwrite(dest + image[:-4] + '_flipud' + '.tif', flipud)

    cropped = tf.image.central_crop(img_load, central_fraction=0.8)
    crop = np.asarray(cropped)
    cv2.imwrite(dest + image[:-4] + '_crop' + '.tif', crop)
```

```python
src = '/content/gdrive/My Drive/Dataset/FL/'

if __name__ == '__main__':
  augmentation(src)
```

Figure 6: **Function for data augmentation**

Histogram Normalization

```python
def hist0(fileName):

    img = Image.open(fileName)

    img = np.asarray(img)

    flat = img.flatten()


    hist = get_histogram(flat, 256)

    #execute the fn
    cs = cumsum(hist)

    # numerator & denomenator
    nj = (cs - cs.min()) * 255
    N = cs.max() - cs.min()

    # re-normalize the cdf
    cs = nj / N
    cs = cs.astype('uint8')
    img_new = cs[flat]
    # put array back into original shape since we flattened it
    img_new = np.reshape(img_new, img.shape)
    return img_new
```

Figure 7: **Function for Histogram Normalization**

is written which will take all the images that need to be normalized and will write all processed images into destination folder with for loop.

The images were accessed with the help of ImageDataGenerator. The images were loaded into training and testing with ImageDataGenerator as shown below.

```
train = train_gen.flow_from_directory("/content/gdrive/My Drive/Augmented_Normalized/train/",
                                      class_mode="categorical",
                                      target_size=(224, 224),
                                      color_mode="rgb",
                                      shuffle=True,
                                      batch_size=32)
```

Figure 8: **Loading Images with ImageDataGenerator**

5.2 Execution of CNN and Transfer Learning

All the required libraries for execution of CNN and trasnfer learning with Inception_v3 and DenseNet121 are shown in figure below. The Figure 9 demonstrates the libraries required for CNN execution.

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense,Reshape
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.utils import np_utils
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from keras.preprocessing import image
from sklearn.preprocessing import MultiLabelBinarizer
import matplotlib.pyplot as plt
import os
import tensorflow as tf
from pathlib import Path
from keras.applications import densenet
from keras.models import Sequential, Model, load_model
from keras.preprocessing import image
import keras
import tensorflow as tf
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, Conv3D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from keras.optimizers import Adam
from keras.losses import mae, sparse_categorical_crossentropy, binary_crossentropy , categorical_crossentropy
```

Figure 9: **Required libraries for CNN**

The Figure 10 demonstrates the architecture of CNN used in this research project. For execution of transfer learning, the libraries shown in Figure 11 are imported.

```
model = Sequential()
model.add(Conv2D(filters = 16, kernel_size = 3, padding = 'same', activation = 'relu', input_shape = (224, 224, 3)))
model.add(Dropout(0.3))
model.add(MaxPooling2D(pool_size = 3))

model.add(Conv2D(filters = 32, kernel_size = 3, padding = 'same', activation = 'relu'))
model.add(Dropout(0.3))
model.add(MaxPooling2D(pool_size = 3))

model.add(Conv2D(filters = 64, kernel_size = 3, padding = 'same', activation = 'relu'))
model.add(Dropout(0.3))
model.add(MaxPooling2D(pool_size = 3))

model.add(Conv2D(filters = 128, kernel_size = 3, padding = 'same', activation = 'relu'))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(3, activation = 'softmax'))

model.compile(optimizer=Adam(0.0001), loss=categorical_crossentropy, metrics=['accuracy'])
model.summary()
return model
```

Figure 10: **CNN Model**

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense,Reshape
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.utils import np_utils
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
from keras.applications import densenet
from keras.models import Sequential, Model, load_model
from keras.preprocessing import image
import seaborn as sn
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from keras.application import DenseNet121
from tensorflow.keras import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow import keras
```

Figure 11: **libraries for DenseNet121 and Inception_v3**

5.3 Training and Validation Accuracy Plot

The training and validation graph was plotted with the help of code shown in Figure 12.

```
[ ] # plot the model loss and accuracy
    train_loss = model_history.history['loss']
    train_acc = model_history.history['accuracy']

    valid_loss = model_history.history['val_loss']
    valid_acc = model_history.history['val_accuracy']

    x = [(i+1) for i in range(len(train_loss))]

    f,ax = plt.subplots(1,2, figsize=(12,5))
    ax[0].plot(x, train_loss)
    ax[0].plot(x, valid_loss)
    ax[0].set_title("Loss plot")
    ax[0].set_xlabel("Epochs")
    ax[0].set_ylabel("loss")
    ax[0].legend(['train', 'valid'])


    ax[1].plot(x, train_acc)
    ax[1].plot(x, valid_acc)
    ax[1].set_title("Accuracy plot")
    ax[1].set_xlabel("Epochs")
    ax[1].set_ylabel("acc")
    ax[1].legend(['train', 'valid'])

    plt.show()
```

Figure 12: **Code for training and validation accuracy**

6 Other Software Used

The documentation of the research finding is done with the help of overleaf. The Figure 13 demonstrates how overleaf is used for project documentation.

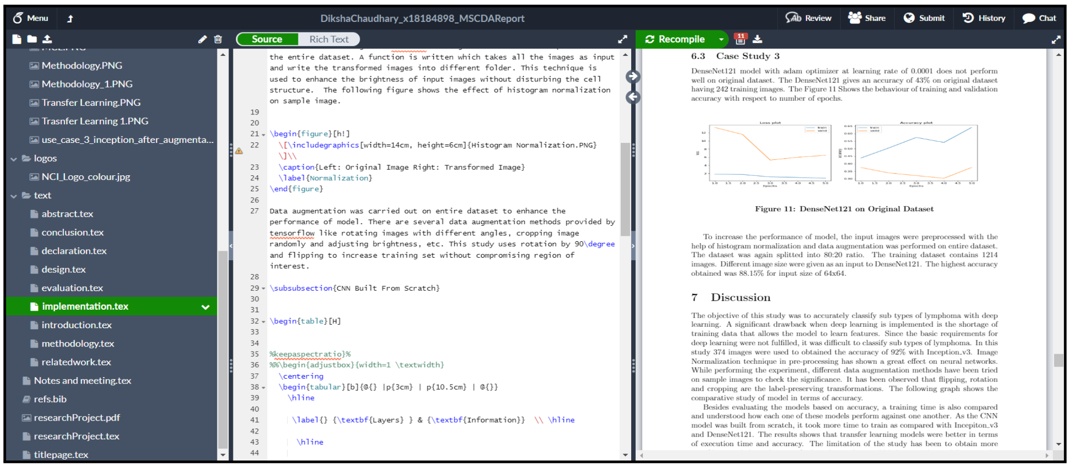

Figure 13: **Overleaf Code**

References

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

https://www.tensorflow.org/tutorials/images/data_augmentation

https://keras.io/api/preprocessing/image/

https://www.tensorflow.org/tutorials/keras/classification

https://matplotlib.org/tutorials/introductory/sample_plots.html