National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Manoj Shukla
Student ID: x18195881

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Manoj Kumar Shukla

Student ID: X18195881

Programme: Data Analytics **Year:** 2020

Module: Research

Lecturer:

Submission Due Date:

Project Title: Glaucoma stages classification using deep learning techniques.

Word Count: 967 **Page Count:** 11

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: Manoj Kumar Shukla

Date: 19-08-2020

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

Manoj Shukla
Student ID: x18195881

1 Introduction

The configuration manual will support to run the research project "Classification of different stages of glaucoma using Deep learning techniques " from the beginning. The configuration manual provides a comprehensive understanding of prerequisites to set up the project and run successfully and examine these results.

This manual is divided into the following segments: section 1 Environment setup details, Section 2 The library loading requirements, section 3 Details about the data and at the end section 4 dataset load 5 Loading data and Segmentation andCDR calculation, and at the end 6 step is how the models are executed.

2 Hardware Specification

- MacBook Pro (15 inches, 2018)
- Processor 2.2 GHz 6-Core Intel Core i7
- Memory: 16GB 2400 MHz DDR4
- Startup Disk: Machintosh HD
- Graphics Intel UHD Graphics 630 1536MB

2.1 Software Specification

Anaconda Navigator for Windows (Version 1.9.7)

- Google collab Pro
- Spyder

2.2 Programming Requisites

- Python (Version 3.7.5)

Google colab is coming preinstall all the requirements, if needed required libs can be installed in runtime.

Command to install required libs in google colab.
! pip install __libName__

3 Libraries required

All the used libraries were needed to build this research project are mentioned blow.

4 Dataset

Two datasets were used in this project, The DRISHTI dataset was used to calculate the Cup to disc ratio (CDR) which helps build segmentation solutions with an accuracy of 85%. Further, this model was used to calculate the CDR value of MESSIDOR data. data can be accessed from? DRISHTI- http://cvit.iiit.ac.in/projects/mip/drishti-gs/mip-dataset2/Home.php MESSIDOR:- https://deepblue.lib.umich.edu/data/concern/data_sets/3b591905z?locale=en Dataset used in this research is open in public for research purposes.
In the Drishti dataset total of 200 images were available. In the MESSIDOR dataset, 3220 TIFF images were present, for the research purpose TIFF images were converted into JPG format, as the first step of the project.

5 Loading Data & Segmentation

DRISHTI dataset is downloaded from the source and segmentation is performed in local machine with the help of Anaconda Spyder. Result can be seen in Fig 1. After getting result further step was to apply this modelling on MESSIDOR data to calculate CDR and categories them into (mild/severe).

MESSIDOR images were converted from TIFF to jpeg Fig 1.

```
import os
from PIL import Image

yourpath = os.getcwd()+'/MESSIDOR'

for root, dirs, files in os.walk(yourpath, topdown=False):
    for name in files:
        print(os.path.join(root, name))
        if os.path.splitext(os.path.join(root, name))[1].lower() == ".tif":
            if os.path.isfile(os.path.splitext(os.path.join(root, name))[0] + ".jpg"):
                print("A jpeg file already exists for %s" % name)
            # If a jpeg is *NOT* present, create one from the tiff.
            else:
                outfile = os.path.splitext(os.path.join(root, name))[0] + ".jpg"
                try:
                    im = Image.open(os.path.join(root, name))
                    print("Generating jpeg for %s" % name)
                    im.thumbnail(im.size)
                    im.save(outfile, "JPEG", quality=100)
                except (Exception, e):
                    print("Something went wrong")
```

Figure 1 TIFF to JPG Conversion code

After converting data from TIFF to JPG, all the images were stored in cloud storage AWS(S3). To access the data in working environment, a http call was made to the hosted URL to store data into project environment, data will receive in zip format, later python code was written to unzip it and rename all images with count from 0 to 3219.

ATTACH GOOGLE DRIVE AND LOAD DATA FROM S3

```
from zipfile import ZipFile
import os
#connect with Google Drive
def Mount():
  from google.c /content/drive (cmd + click)
  drive.mount('/content/drive')
Mount()
#Load Data from S3
def loadDataFromS3():
  outdir = '/tmp/data'
  if not os.path.exists(outdir):
    os.mkdir(outdir)
  !wget --no-check-certificate \
  https://nci-thesis-glaucoma-optic-images-manoj-shukla.s3-eu-west-1.amazonaws.com/MESIDOR_JPG.zip \
  -O /tmp/data/MESIDOR_JPG.zip

  for i, filename in enumerate(os.listdir('/tmp/data/MESIDOR_JPG/')):
    os.rename("/tmp/data/MESIDOR_JPG/" + filename, "/tmp/data/MESIDOR_JPG/image" + str(i) + ".jpg")

#Unzip Data
def unzipData():
  with ZipFile('/tmp/data/MESIDOR_JPG.zip', 'r') as zipObj:
    zipObj.extractall('/tmp/data/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.

Enter your authorization code:
··········
Mounted at /content/drive
```

Figure 2 Load Dataset from AWS S3 Bucket

After images get download and unzip Fig 2. Segmentation is performed on MESSIDOR dataset and calculated value are saved in xl file.

Segementation To Compute CDR Ratio

Load Libs for Segmentations and CDR values

```
from scipy import signal
import cv2
import sys
import glob
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
import os
from PIL import Image
import xlrd
import math
from pylab import*
from scipy import signal
import numpy as np
from matplotlib import pyplot as plt
from skimage.feature import greycomatrix, greycoprops
from skimage import data
```

Figure 3 Libs required to perform segmentation

▾ Segmentation Class

```
class segmentation:
  def __init__(self):
    pass

  def load_image(self, path):
    return np.asarray(Image.open(path))# returns an image of dtype int in range [0, 255]

  def load_set(self, folder, shuffle=False):
      img_list = sorted(glob.glob(os.path.join(folder, '*.png')) + \
                        glob.glob(os.path.join(folder, '*.jpg')) + \
                        glob.glob(os.path.join(folder, '*.jpeg')))
      if shuffle:
          np.random.shuffle(img_list)
      data = []
      filenames = []
      for img_fn in img_list:
          img = self.load_image(img_fn)
          data.append(img)
          filenames.append(img_fn)
      return data, filenames

  def getDataSet(self, db_folder,cdr,train_data):
    file_codes_all = []
    if train_data:
        set_path = os.path.join(db_folder, 'MESIDOR_JPG')
    else:
        set_path = os.path.join(db_folder,'MESIDOR_JPG')
    images_path = os.path.join(set_path)
    X_all, file_names = self.load_set(images_path)
    rel_file_names = [os.path.split(fn)[-1] for fn in file_names]
    rel_file_names_wo_ext = [fn[:fn.rfind('.')] for fn in rel_file_names]
    if train_data:
        file_codes = [fn[fn.find('_'):] for fn in rel_file_names_wo_ext]
    else:
        file_codes = [fn[fn.find('_'):] for fn in rel_file_names_wo_ext]
    file_codes_all.extend(file_codes)
    return X_all, file_codes_all, file_names

  def Segment(self, image,plot_seg,plot_hist):

    image = image[400:1400,500:1600,:] #cropping the fundus image to ger region of interest

    Abo,Ago,Aro = cv2.split(image)  #splitting into 3 channels
    #Aro = clahe.apply(Aro)
    Ago = clahe.apply(Ago)
    M = 60    #filter size
    filter = signal.gaussian(M, std=6) #Gaussian Window
    filter=filter/sum(filter)
    STDf = filter.std()  #It'standard deviation


    Ar = Aro - Aro.mean() - Aro.std() #Preprocessing Red
```

Figure 4 Segmentation Code

After Segmentation CDR ratio were calculated using python Fig 5 in google colab, using MESSIDOR data set.

Calculate CDR

```
def CalculateCDR(cup,disc,plot):

    #morphological closing and opening operations
    R1 = cv2.morphologyEx(cup, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(2,2)), iterations = 1)
    r1 = cv2.morphologyEx(R1, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(7,7)), iterations = 1)
    R2 = cv2.morphologyEx(r1, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(1,21)), iterations = 1)
    r2 = cv2.morphologyEx(R2, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(21,1)), iterations = 1)
    R3 = cv2.morphologyEx(r2, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(33,33)), iterations = 1)
    r3 = cv2.morphologyEx(R3, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(43,43)), iterations = 1)

    img = clahe.apply(r3)


    ret,thresh = cv2.threshold(cup,127,255,0)
    #img,
    contours,hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE) #Getting all possible contours in
    cup_diameter = 0
    largest_area = 0
    el_cup = contours[0]
    if len(contours) != 0:
        for i in range(len(contours)):
            if len(contours[i]) >= 5:
                area = cv2.contourArea(contours[i]) #Getting the contour with the largest area
                if (area>largest_area):
                    largest_area=area
                    index = i
                    el_cup = cv2.fitEllipse(contours[i])

    cv2.ellipse(img,el_cup,(140,60,150),3)  #fitting ellipse with the largest area
    x,y,w,h = cv2.boundingRect(contours[index]) #fitting a rectangle on the ellipse to get the length of major axis
    cup_diameter = max(w,h) #major axis is the diameter

    #morphological closing and opening operations
    R1 = cv2.morphologyEx(disc, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(2,2)), iterations = 1)
    r1 = cv2.morphologyEx(R1, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(7,7)), iterations = 1)
    R2 = cv2.morphologyEx(r1, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(1,21)), iterations = 1)
    r2 = cv2.morphologyEx(R2, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(21,1)), iterations = 1)
    R3 = cv2.morphologyEx(r2, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(33,33)), iterations = 1)
    r3 = cv2.morphologyEx(R3, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(43,43)), iterations = 1)

    img2 = clahe.apply(r3)

    ret,thresh = cv2.threshold(disc,127,255,0)
    #img2,
    contours,hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE) #Getting all possible contours in
    disk_diameter = 0
    largest_area = 0
    el_disc = el_cup
    if len(contours) != 0:
          for i in range(len(contours)):
            if len(contours[i]) >= 5:
                area = cv2.contourArea(contours[i]) #Getting the contour with the largest area
```

Figure 5 Calculation of Cup to Disk Ratio on MESSIDOR Data

6 classification Model Implementation

6.1 VGG16

Implemented Process of VGG16 and used fine-tuned model Fig 6 to train the model, using Keras, TensorFlow and python.

VGG16

```
def createVGG16Model():
  vgg16_model = keras.applications.vgg16.VGG16()
  vgg16_model.summary()
  model = Sequential()
  for layer in vgg16_model.layers[:-1]:
      model.add(layer)
  model.summary()
  for layer in model.layers:
    layer.trainable = False
  model.add(Dense(2, activation='softmax'))
  model.summary()
  model.compile(SGD(lr=.0005), loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

Figure 6 Model Codes

6.2 VGG16

Implemented Process of VGG19 and used fine-tuned model Fig 7 to train the model, using Keras, TensorFlow and python.

VGG19

```
def createVGG19Model():
  vgg19_model = keras.applications.vgg19.VGG19()
  vgg19_model.summary()
  model = Sequential()
  for layer in vgg19_model.layers[:-1]:
      model.add(layer)
  model.summary()
  for layer in model.layers:
    layer.trainable = False
  model.add(Dense(2, activation='softmax'))
  model.summary()
  model.compile(Adam(lr=.0005), loss='binary_categorical_crossentropy', metrics=['accuracy'])
  return model
```

Figure 7 VGG 19 Model Codes

6.3 ResNet50

Implemented Process of VGG19 and used model with custom layers. Fig 8 to train the model, using Keras, TensorFlow and python.

ResNet50

```
[ ]  1 from tensorflow.keras.applications import ResNet50
     2 from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D, BatchNormalization
     3 from tensorflow.keras.applications.resnet50 import preprocess_input
     4 from keras import regularizers
```

```
[ ]  1 def createResNet50Model():
     2   model = Sequential()
     3   model.add(ResNet50(include_top=False, pooling='avg', weights='imagenet'))
     4   model.add(Flatten())
     5   model.add(BatchNormalization())
     6   model.add(Dense(2048, activation='relu'))
     7   model.add(BatchNormalization())
     8   model.add(Dense(1024, activation='relu'))
     9   model.add(BatchNormalization())
    10   model.add(Dense(2, activation='softmax'))
    11   # for layer in model.layers:
    12   #     layer.trainable = False
    13   model.layers[0].trainable = False
    14   model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    15   return model
```

```
[ ]  1 resnet50_epochs = 50
     2 train_batches, valid_batches, test_batches = createBatch(train_path, valid_path, test_path, True)
     3 resNet50Model = createResNet50Model()
     4 resNet50_history = TrainModel(resNet50Model, train_batches, valid_batches, resnet50_epochs)
     5 ConfusionMatrics(resNet50Model, test_batches, resNet50_history)
     6
```

Figure 8 ResNet50 Model Codes

6.4 IncptionV3

Implemented Process of InceptionV3 and used model with custom layers. Fig 9 to train the model, using Keras, TensorFlow and python.

InceptionV3

```
 1 def createInceptionV3Model():
 2   from tensorflow.keras.applications import InceptionV3
 3   opt = Adam(learning_rate=0.001)
 4   base_model = InceptionV3(include_top=False, pooling='avg', weights='imagenet')
 5   model = Sequential()
 6   model.add(base_model)
 7   model.add(Flatten())
 8   model.add(BatchNormalization())
 9   model.add(Dense(2048, activation='relu'))
10   model.add(BatchNormalization())
11   model.add(Dense(1024, activation='relu'))
12   model.add(BatchNormalization())
13   model.add(Dense(64, input_dim=64, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
14   model.add(BatchNormalization())
15   model.add(Dense(2, activation='softmax'))
16   model.layers[0].trainable = False
17   model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
18   return model
19
20
```

```
[ ]  1 incepV3_epochs = 50
     2 train_batches, valid_batches, test_batches = createBatch(train_path, valid_path, test_path, False)
     3 incepV3Model = createInceptionV3Model()
     4 incepV3_history = TrainModel(incepV3Model,train_batches, valid_batches, incepV3_epochs)
     5 ConfusionMatrics(incepV3Model, test_batches, incepV3_history)
```

Found 605 images belonging to 2 classes.

Figure 9 InceptionV3 Model codes

6.5 IncptiionResNetV2

Implemented Process of InceptionResNetV2 and used model with custom layers. Fig 10 to train the model, using Keras, TensorFlow and python.

▾ InceptionResNetV2

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from tensorflow.keras.applications import InceptionResNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D, BatchNormalization
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from keras.layers import Dropout
from keras import regularizers
from keras.layers import Dropout
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.layers import GaussianNoise
import keras
import cv2
```

```
[ ]
def createDenseNet50(input_shape=(229,229,3), oclass=2):
  model = Sequential()
  model.add(InceptionResNetV2(include_top=False, pooling='avg', weights='imagenet'))
  model.add(Flatten())
  model.add(BatchNormalization())
  model.add(Dense(2048, activation='relu'))
  model.add(BatchNormalization())
  model.add(Dropout(0.2))
  model.add(Dense(1024, activation='relu'))
  model.add(BatchNormalization())
  model.add(Dropout(0.2))
  model.add(Dense(64, input_dim=64, kernel_regularizer=regularizers.l2(0.0001)))
  model.add(BatchNormalization())
  model.add(Dropout(0.3))
  model.add(Dense(2, activation='softmax'))

  for layer in model.layers:
    layer.trainable = False
  for i in range(-9, 0):
    model.layers[i].trainable = True
  return model
```

Figure 10 InceptionResNetV2 Model codes

6.6 DenseNet169

Implemented Process of InceptionResNetV2 and used model with custom layers. Fig 11 to train the model, using Keras, TensorFlow and python.

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from tensorflow.keras.applications import DenseNet169
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D, BatchNormalization
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from keras.layers import Dropout
from keras import regularizers
from keras.layers import Dropout
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.layers import GaussianNoise
import keras
import cv2
```

```
def createDenseNet50():
  model = Sequential()
  model.add(DenseNet169(include_top=False, pooling='avg', weights='imagenet'))
  model.add(Flatten())
  model.add(BatchNormalization())
  model.add(Dense(2048, activation='relu'))
  model.add(BatchNormalization())
  model.add(Dense(1024, activation='relu'))
  model.add(BatchNormalization())
  model.add(Dense(64, input_dim=64, kernel_regularizer=regularizers.l2(0.0001)))
  model.add(BatchNormalization())
  model.add(Dropout(0.2))
  model.add(Dense(2, activation='softmax'))

  for layer in model.layers:
    layer.trainable = False
  for i in range(-9, 0):
    model.layers[i].trainable = True
  return model
```

```
denseNet_epochs = 20
train_batches, valid_batches, test_batches = createBatch(train_path, valid_path, test_path, False, 32, (229,229))
inceptionResnetV2Model = createDenseNet50()
inceptionResnetV2Model.compile(optimizer=Adam(lr=1e-5), loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 11 DenseNet169 Model Codes

References

- Code Reference CDR Calculation https://github.com/NupurBhaisare/Cup-and-disc-segmentation-for-glaucoma-detection-CDR-Calculation.

- Code Reference for Tensorflow Learning: https://www.tensorflow.org/api docs/python/tf

- Code Reference for Learning Keras : https://www.tensorflow.org/api docs/python/tf/keras/