National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Nandita Sharma
Student ID: x18185100

School of Computing
National College of Ireland

Supervisor: Dr. Rashmi Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Nandita Sharma……

Student ID: …x18185100……

Programme: …Msc in Data Analytics…… **Year:** 2019-2020…..

Module: …Msc in Research Project……

Lecturer: ……

Submission Due Date: ……17 August 2020……

Project Title: ..”Optic Disc and Cup Segmentation in Glaucoma Screening using Mask RCNN “…

Word Count: …1489…… **Page Count:** ……20……

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: ……*Nandita Sharma*……

Date: …17 August 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

Nandita Sharma
X18185100

Introduction

The Configuration manual represents the various parameter and configuration used to perform this research like installation and requirements. The stepwise explanation to execute the program has been explained in this manual.

1 Environment Specification and Configuration

1.1 Hardware Configuration

The screenshot of hardware configuration of system details in figure 1 can be seen.

- Windows Edition: Windows 10 Pro edition.
- Processor: Intel(R) Core™ i5-4210U CPU @ 1.70GHz 2.40 GHz
- Installed Memory (RAM) : 8GB
- System type: 64-bit operating System, x64-based processor
- Pen and Touch: no pen or touch input is available

Figure 1

1.2 Software Configuration

The Anaconda is installed and then IDE such as Jupyter notebook is installed.
The steps to installs anaconda is shown below through figures:

1.2.1 Anaconda installation

Steps 1: Go to website: Anaconda.com/downloads
Page will look something like this: Figure 2.

Figure 2

Steps 2: If you using windows click windows or depending on your operating system. The proposed research choose windows option shown in below figure 3.

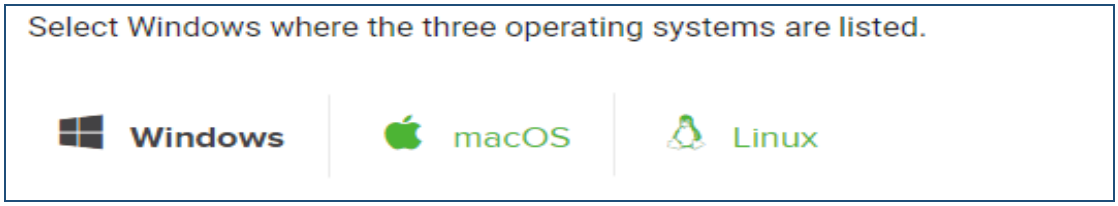

Figure 3

Steps 3: The latest Version of python 3.6 is installed supporting 64-bit graphical installer.

Figure 4

Step 4: After downloading the install exe file and follow steps shown in the below images:

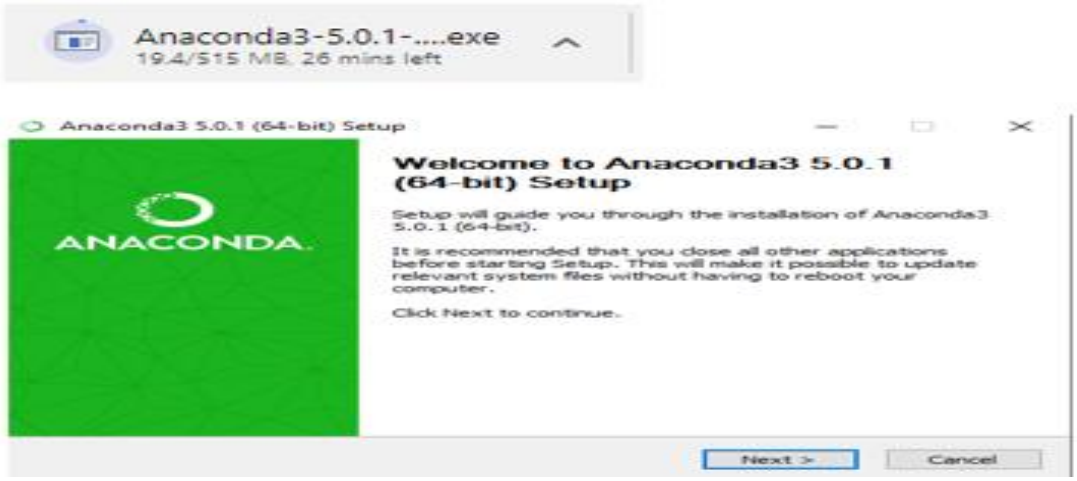

Figure 5

Step 5: Agree with license and click install. Figure 6.

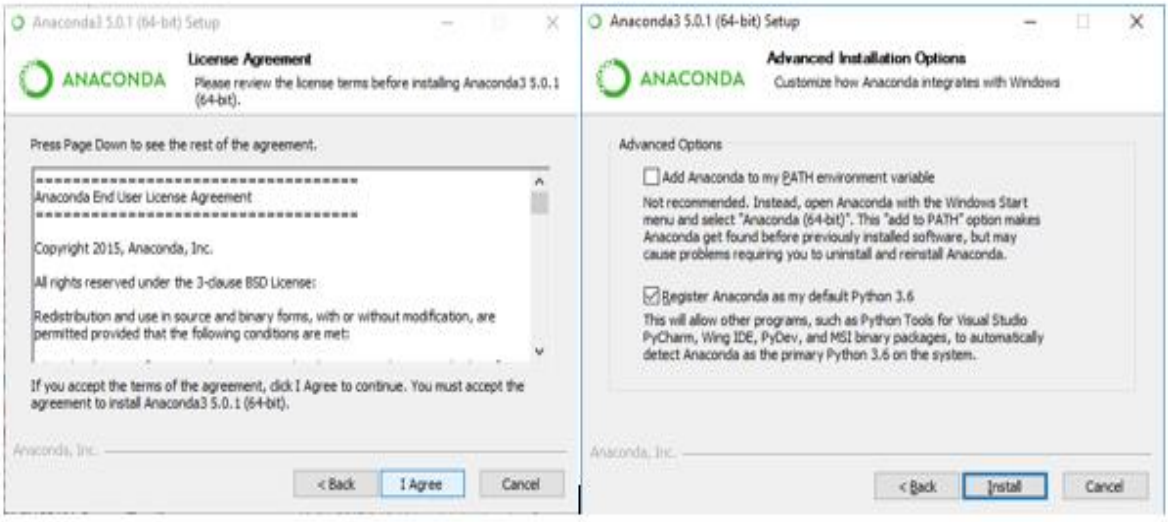

Figure 6

Step 6: Now open anaconda

1.2.2 Juypter Installation

Step 1: From anaconda launch “Jupyter”. There are various IDE can be seen in Anaconda Navigator. Depending on use case any IDE can be used.
Notebook shown in figure **7.**

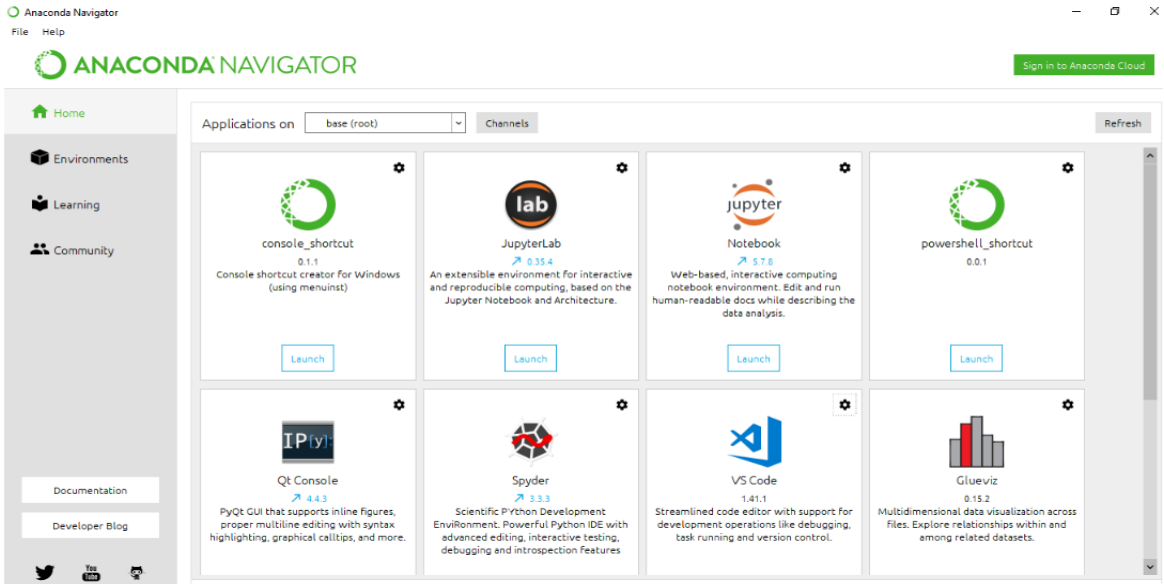

Figure 7

1.2.3 Google Colab:

Google colab is used for training the model. It is a free cloud service. Colab is just like a jupyter notebook as it requires the first connection to be made with drives by assigning a folder path to it. One can change runtime by selecting a GPU or TPU. It provides 13 GB of RAM along with GPU and TPU support free.

Step 1: From google drive > go to app > click right > More > open colaboratory
The page will open appears as figure 8 and then Rename it with your project name.

Figure 8

Step 2: Go in "Runtime" in above figure > click change runtime > select GPU > Save Refer Figure 9.

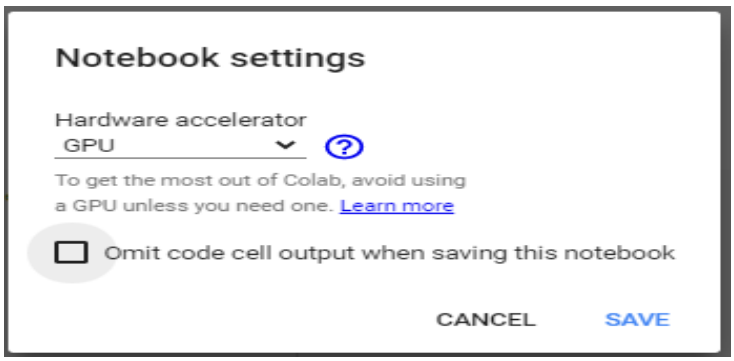

Figure 9

2 Data Pre-processing

The pre-processing part is done in Jupyter notebook.

2.1 Import all python libraries required for data pre-processing. Code 1.

```
import os
import cv2
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
import pandas as pd
import imutils
```

Code 1

2.2 Image Resizing is done to bring all images of one size.

Below code 2 shows code for image size 512 x 512.

```
import os
import cv2
import numpy as np
from tqdm import tqdm
path =["E:\\Dataset\\work2\\BinRushed4"]
destination =[" E:\\Dataset\\work2\\Glaucoma_final"]
image = [ ]
for n, i in enumerate (path):
    for j in tqdm (os.listdir(i)):
    print(j)
    img_path =  os.path.join(i,j)
    img = cv2.resize(img, (512,512))
    cv2.imwrite(os.path.join(destination[0],j), img2)
```

Code 2

2.3 Image Enhancement

Refer Code 3 for image enhancement

```
import shutil
import os
from PIL import Image
from PIL import ImageFilter
source = "E:\\Glacoma\\"
dest2 = "E:\\Sharp\\"
for filename in os.listdir(source):
      if filename.endswith(".JPG"):
           Path_image=os.path.join(source, filename)
           Print(Path_image)
           dgeEnhanced = img.filter(ImageFilter.EDGE_ENHANCE)
           dgeEnhanced.save(dest2+filename)
```

Code 3

3 Download Dataset

The dataset is downloaded from “Deep blue data repository”, from link https://deepblue.lib.umich.edu/data/concern/data_sets/3b591905z. (1)
The Dataset name is “the RIGA dataset”.

3.1 Annotate Data

The fundus image is labelled by six experienced pathologists. The image is annotated using “LabelMe tool” on labelled mask provided by pathologist. (Marois and Syssau, 2006) proposed LabelMe tool. Figure 10 is LabelMe tool.

Step 1: Open Anaconda cmd and write below code shown in Code 4 and install “LabelMe tool”

```
conda create  --name=labelme python=3.6
source activate labelme
pip install labelme
```

Code 4

Step 2: Open labelme > open dir > choose folder where image is present > from left tool bar > select polygon > start Annotate > Save as a label name for object.

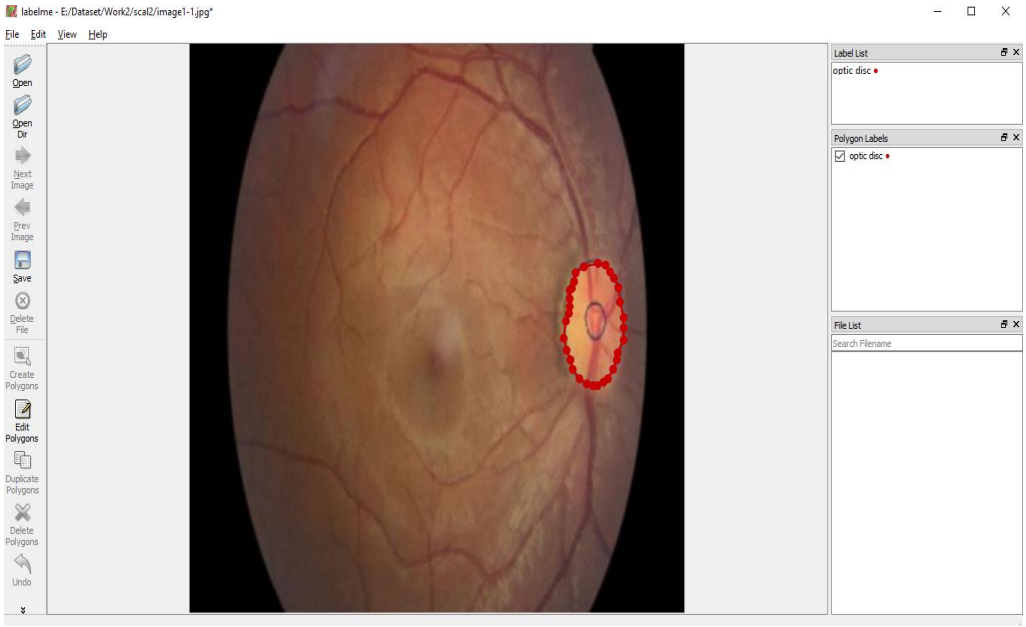

Figure 10

4 Prepare setup for data modelling

4.1 Upload Dataset to drive

Create a main folder name “Project” and subfolder name “Dataset”. Upload complete dataset containing fundus image and annotated files to google drive in “Dataset folder”. And also create colab notebook under the project folder. It can be seen in figure 11.

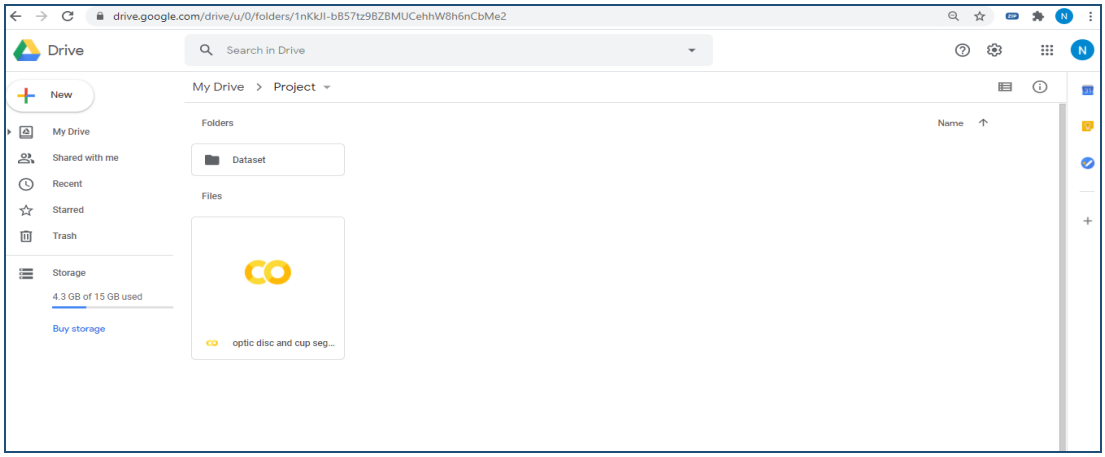

Figure 11

4.2 Connection from drive to colab notebook

Below code is used to connect drive data to colab notebook shown in Code 5.

```
from google.colab import drive
drive.mount ('/content/drive')
```

Code 5

The authorization code can be obtained by clicking “https” link. The figure 12 will get open in another tab, copy the key link and paste it in output of code 5.

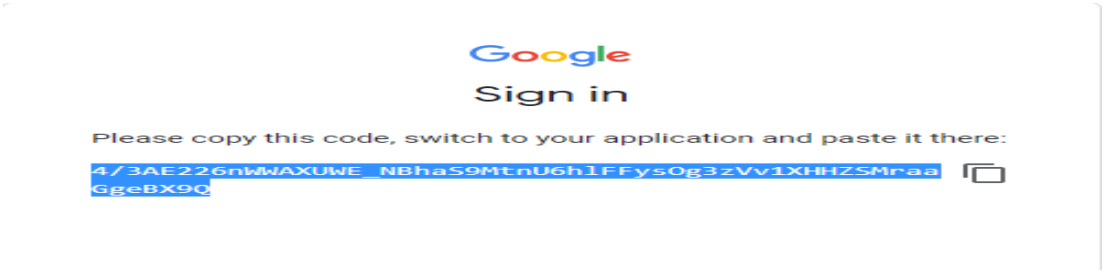

Figure 12

4.3 Import Keras and Tensorflow

Install Keras version 2.1.0 and Tensorflow version 1x. This version is supported for Mask RCNN. Refer code 6.

```
%tensorflow_version 1.x
!pip3 uninstall -y keras
!pip3 install keras == 2.1.0
```

Code 6

4.4 Import Package of model Mask R-CNN

(He *et al.*, 2017) proposed deep neural network model "Mask RCNN" that can perform accurate segmentation and detection. Thus, Mask RCNN is used in this research. From the below code, clone Mask RCNN supporting files from official repository. Refer code 7 for same.

```
!git clone https://github.com/matterport/Mask_RCNN
%cd Mask_RCNN
!pip3 install -r requirements.txt
!python3 setup.py install
```

Code 7

Also clone Cocoapi weight from official repository as shown in code no 8.

```
!git clone https://github.com/cocodataset/cocoapi.git
%cd cocoapi/PythonAPI
!make
%cd ../../
```

Code 8

4.5 Import libraries and set root directory and import supporting py files for Mask RCNN

The import "visualize", "utils" from "mrcnn". Import "log" from "mrcnn.model" and import function "config" from "mrcnn.config". First pre-trained coco weight is used to train the model. Thus "ROOT_DIR" is joined with pre-trained weight "mask_rcnn_coco.h5" can be seen in code 9.

```
import os
import sys
import random
import math
import re
import time
import numpy as np
import cv2
import matplotlib
import matplotlib.pyplot as plt
import json
#ROOT_DIR = os.path.abspath("../")
ROOT_DIR = os.path.abspath('/content/drive/My Drive/Project_try/Mask_RCNN/')
# Import Mask RCNN
```

```
sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn.config import Config
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize
from mrcnn.model import log
%matplotlib inline
# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")
# Local path to trained weights file
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
# Download COCO trained weights from Releases if needed
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)
```

Code 9

4.6 Create Main "Config"function

```
class GlaucomaCupConfig(Config):
    # Give the configuration a recognizable name
    NAME = "Glaucoma_cup"
    NUM_CLASSES = 1 + 2
    STEPS_PER_EPOCH = 200
    VALIDATION_STEPS = 50
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    IMAGE_MIN_DIM =  512
    IMAGE_MAX_DIM =  512
    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
    TRAIN_ROIS_PER_IMAGE = 32
    MAX_GT_INSTANCES = 50
    POST_NMS_ROIS_INFERENCE = 500
    POST_NMS_ROIS_TRAINING = 1000
config = GlaucomaCupConfig()
config.display()
```

Code 10

Code 10 shown "config" class named "Glaucomacupconfig". There are various parameter could be change depending upon the project use case.

4.7 Load image Data with annotate file and add class

Code 11 is a code that connects data with annotate file which is in "JSON" format. And label, number of classes name such as "optic cup" and "optic disc" present in dataset.

```
class GlaucomaDataset(utils.Dataset):
    def load_dataset(self, dataset_dir):
        self.add_class('dataset', 1, 'optic cup')
        self.add_class('dataset', 2, 'optic disc')
        # find all images
        for i, filename in enumerate(os.listdir(dataset_dir)):
            if '.jpg' in filename:
                self.add_image('dataset',
                               image_id=i,
                               path=os.path.join(dataset_dir, filename),
                               annotation=os.path.join(dataset_dir,
filename.replace('.jpg', '.json')))
    def extract_masks(self, filename):
```

```
        json_file = os.path.join(filename)
        with open(json_file) as f:
            img_anns = json.load(f)
       masks = np.zeros([512, 512, len(img_anns['shapes'])], dtype='uint8')
       classes = []
        for i, anno in enumerate(img_anns['shapes']):
            mask = np.zeros([512, 512], dtype=np.uint8)
            cv2.fillPoly(mask, np.array([anno['points']], dtype=np.int32), 1)
            masks[:, :, i] = mask
            classes.append(self.class_names.index(anno['label']))
        return masks, classes
    # load the masks for an image
    def load_mask(self, image_id):
        # get details of image
        info = self.image_info[image_id]
        # define box file location
        path = info['annotation']
        # load XML
        masks, classes = self.extract_masks(path)
        return masks, np.asarray(classes, dtype='int32')
    def image_reference(self, image_id):
        info = self.image_info[image_id]
        return info['path']
```

Code 11

4.8 Visualize random sample

```
# Load and display random samples
image_ids = np.random.choice(dataset_train.image_ids, 4)
for image_id in image_ids:
    image = dataset_train.load_image(image_id)
    mask, class_ids = dataset_train.load_mask(image_id)
    visualize.display_top_masks(image, mask, class_ids, dataset_train.class_names)
```

Code 12

4.9 Slip dataset into train and Validation

Train and Validation code can be seen in code 13.

```
# train set
dataset_train = GlaucomaDataset()
dataset_train.load_dataset('/content/drive/My Drive/Project_try/Dataset/Train')
dataset_train.prepare()
# test/val set
dataset_val = GlaucomaDataset()
dataset_val.load_dataset('/content/drive/My Drive/Project_try/Dataset/Val')
dataset_val.prepare()
```

Code 13

5 Create Model

The code 14 shows, model is created in training mode.

```
model = modellib.MaskRCNN(mode="training", config=config,
                          model_dir=MODEL_DIR)
```

Code 14

5.1 Weight to start with

First model is trained on pre-trained coco weight. Refer Code 15.

```
init_with = "coco"  # imagenet, coco, or last
if init_with == "imagenet":
    model.load_weights(model.get_imagenet_weights(), by_name=True)
elif init_with == "coco":
    # Load weights trained on MS COCO, but skip layers that
    # are different due to the different number of classes
    # See README for instructions to download the COCO weights
    model.load_weights(COCO_MODEL_PATH, by_name=True,
                       exclude=["mrcnn_class_logits", "mrcnn_bbox_fc",
                                "mrcnn_bbox", "mrcnn_mask"])
elif init_with == "last":
    # Load the last model you trained and continue training
    model.load_weights(model.find_last(), by_name=True)
```

Code 15

5.2 Train model for 8 epochs

```
model._get_distribution_strategy = lambda: None
model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=8,
            layers='heads',
            #save_each_n_epoch=1,
            custom_callbacks=[mean_average_precision_callback])
```

Code 16

6 Performance Evaluation and Graph

6.1 Plot loss Graph

As model history get stored in “model”. Thus “keras_model.history” is used.

```
hist = model.keras_model.history.history

# Plot of training and validation losses for  mask, bbox, class and total loss
plt.plot(hist['loss'])
plt.plot(hist['val_loss'])
plt.title('Training and Validation loss')
```

```
plt.ylabel('loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(hist['mrcnn_bbox_loss'])
plt.plot(hist['val_mrcnn_bbox_loss'])
plt.title('Training and Validation bounding box loss')
plt.ylabel('loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(hist['mrcnn_mask_loss'])
plt.plot(hist['val_mrcnn_mask_loss'])
plt.title('Training and Validation mask loss')
plt.ylabel('loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

Code 17

6.2 Log Graph

Launch tensorboard graph. The code for log graph is presented in Code 18.

```
!pip install tensorboardcolab
%load_ext tensorboard
import datetime, os
%tensorboard --logdir logs
```

Code 18

6.3 Use model in inference mode

Now trained weight that is obtained at 8th epochs is used in inference mode. Create another "inferenceConfig". Veiw Code 19.

```
class InferenceConfig(GlaucomaCupConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    #IMAGE_MIN_DIM =  512
    #IMAGE_MAX_DIM =  1024
inference_config = InferenceConfig()
# Recreate the model in inference mode
model = modellib.MaskRCNN(mode="inference",
                          config=inference_config,
                          model_dir=MODEL_DIR)
# Get path to saved weights
# Either set a specific path or find last trained weights
# model_path = os.path.join(ROOT_DIR, ".h5 file name here")
model_path = model.find_last()
# Load trained weights
print("Loading weights from ", model_path)
model.load_weights(model_path, by_name=True)
```

Code 19

6.4 Accuracy

Use below code code 20 for calculating accuracy.

```
# Compute VOC-Style mAP @ IoU=0.5
image_ids = dataset_val.image_ids
APs = [ ]
precision = [ ]
recall = [ ]

i =0
for image_id in image_ids:
    # Load image and ground truth data
    image, image_meta, gt_class_id, gt_bbox, gt_mask =\
        modellib.load_image_gt(dataset_val, inference_config,
                               image_id, use_mini_mask=False)
    molded_images = np.expand_dims(modellib.mold_image(image, inference_config), 0)
    # Run object detection
    results = model.detect([image], verbose=0)
    r = results[0]
    # Compute AP
    AP, precisions, recalls, overlaps =\
        utils.compute_ap(gt_bbox, gt_class_id, gt_mask,
                         r["rois"], r["class_ids"], r["scores"], r['masks'])
    APs.append(AP)
    precision.append(precisions)
    recall.append(recalls)
print("mAP: ", np.mean(APs))
```

Code 20

6.5 Confusion Matrix

Code number 21 shows code for confusion matrix to calculate precision, recall, F1 score.

```
gt_tot=gt_tot.astype(int)
pred_tot=pred_tot.astype(int)
#save the vectors of gt and pred
save_dir = "/content/drive/My Drive/Project_try/SaveCM"
gt_pred_tot_json = {"gt_tot" : gt_tot, "pred_tot" : pred_tot}
df = pd.DataFrame(gt_pred_tot_json)
#if not os.path.exists(save_dir):
    #os.makedir(save_dir)

df.to_json(os.path.join(save_dir,"gt_pred_test.json"))

#print the confusion matrix :
print (gt_tot)
print (pred_tot)
y_test = np.array(gt_tot)
predic = np.array(pred_tot)
"""
  Examples to validate output (confusion matrix plot)
    actual: 5 and prediction 1   >>  3
    actual: 2 and prediction 4   >>  1
    actual: 3 and prediction 4   >>  10
"""
annot = True;
cmap = 'Oranges';
fmt = '.2f'
lw = 0.6
cbar = False
show_null_values = 2
```

```
pred_val_axis = 'y'
#size::
fz = 24;
figsize = [36,36];
if(len(y_test) > 10):
    fz=24; figsize=[36,36];
tp, fp, fn = plot_confusion_matrix_from_data(y_test, predic, columns=None,
annot=True, cmap="Oranges",
      fmt='.2f', fz=11, lw=0.5, cbar=False, figsize=[6,6], show_null_values=0,
pred_val_axis='lin')
#tp, fp, fn = plot_confusion_matrix_from_data(y_test, predic, columns = columns,
 #                                            annot, cmap, fmt, fz, lw, cbar,
figsize, show_null_values, pred_val_axis)
#return tp, fp, fn
#tp, fp, fn = utils._test_data_class(dataset_val.class_names, gt_tot, pred_tot)
print()
#ap,mrec,mprec=utils.voc_ap(tp, fp, fn)
ap,mrec,mprec= tp,fp, fn
```

Code 21

6.6 Visualized Predicted region (optic cup and disc). Refer Code 22.

```
image_id = random.choice(dataset_val.image_ids)
image, image_meta, gt_class_id, gt_bbox, gt_mask =\
    modellib.load_image_gt(dataset_val, config, image_id, use_mini_mask=False)
info = dataset_val.image_info[image_id]
print("image ID: {}.{} ({}) {}".format(info["source"], info["id"], image_id,
                                       dataset_val.image_reference(image_id)))
# Run object detection
results = model.detect([image], verbose=1)
# Display results
ax = get_ax(1)
r = results[0]
visualize.display_differences(image, gt_bbox, gt_class_id, gt_mask, r['rois'],
r['class_ids'],r['scores'],r['masks'],dataset_val.class_names,ax=ax)
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)
```

Code 22

7 Detected region

7.1 Detect bounding box from fundus image. Code 23 for reference.

```
# Load random image and mask.
image_id = random.choice(dataset.image_ids)
image = dataset.load_image(image_id)
mask, class_ids = dataset.load_mask(image_id)
# Compute Bounding box
bbox = utils.extract_bboxes(mask)
# Display image and additional stats
print("image_id ", image_id, dataset.image_reference(image_id))
log("image", image)
log("mask", mask)
log("class_ids", class_ids)
log("bbox", bbox)
# Display image and instances
visualize.display_instances(image, bbox, mask, class_ids, dataset.class_names)
```

Code 23

.

7.2 Mini Mask

Mini mask of region can be observed through code in Code 24.

```
from mrcnn.visualize import display_images
image_id = np.random.choice(dataset.image_ids, 1)[0]
image, image_meta, class_ids, bbox, mask = modellib.load_image_gt(
    dataset, config, image_id, use_mini_mask=False)
log("image", image)
log("image_meta", image_meta)
log("class_ids", class_ids)
log("bbox", bbox)
log("mask", mask)
display_images([image]+[mask[:,:,i] for i in range(min(mask.shape[-1], 7))])
```

Code 24

7.3 Mask Resizing.

Mask resizing code can be seen through Code 25.

```
# Add augmentation and mask resizing.
image, image_meta, class_ids, bbox, mask = modellib.load_image_gt(
    dataset, config, image_id, augment=True, use_mini_mask=True)
log("mask", mask)
display_images([image]+[mask[:,:,i] for i in range(min(mask.shape[-1], 7))])
```

Code 25

7.4 Detect formed Anchor tags

The anchor tags code is shown in Code 26 below:

```
# Generate Anchors
backbone_shapes = modellib.compute_backbone_shapes(config, config.IMAGE_SHAPE)
anchors = utils.generate_pyramid_anchors(config.RPN_ANCHOR_SCALES,
                                         config.RPN_ANCHOR_RATIOS,
backbone_shapes, config.BACKBONE_STRIDES,  config.RPN_ANCHOR_STRIDE )
# Print summary of anchors
num_levels = len(backbone_shapes)
anchors_per_cell = len(config.RPN_ANCHOR_RATIOS)
print("Count: ", anchors.shape[0])
print("Scales: ", config.RPN_ANCHOR_SCALES)
print("ratios: ", config.RPN_ANCHOR_RATIOS)
print("Anchors per Cell: ", anchors_per_cell)
print("Levels: ", num_levels)
anchors_per_level = []
for l in range(num_levels):
    num_cells = backbone_shapes[l][0] * backbone_shapes[l][1]
    anchors_per_level.append(anchors_per_cell * num_cells //
config.RPN_ANCHOR_STRIDE**2)
    print("Anchors in Level {}: {}".format(l, anchors_per_level[l]))
```

References

He, K. *et al.* (2020) 'Mask R-CNN', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2), pp. 386–397. doi: 10.1109/TPAMI.2018.2844175.
Marois, B. and Syssau, P. (2006) 'Pratiques des banques françaises en termes d'analyse du risque-pays', *Revue Francaise de Gestion*, 162(3), pp. 77–91. doi: 10.3166/rfg.162.77-94.

Deep Blue Data repository. (2020). https://deepblue.lib.umich.edu/data/about-top, Regents of the University of Michigan.

Marois, B. and Syssau, P. (2006) 'Pratiques des banques françaises en termes d'analyse du risque-pays', *Revue Francaise de Gestion*, 162(3), pp. 77–91. doi: 10.3166/rfg.162.77-94.