Configuration Manual

MSc Research Project
Data Analytics

Ravjyot Singh Duggal
Student ID: x21128901

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ravjyot Singh Duggal
Student ID:	x21128901
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr. Christian Horn
Submission Due Date:	15-12-22
Project Title:	Deep Learning for Driver Drowsiness Detection
Word Count:	646
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:	Ravjyot Singh Duggal
Date:	15th December 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

Ravjyot Singh Duggal
x21128901

1 Introduction

This document contains the instructions for reproducing the code for the project on driver drowsiness. The steps taken for reproducing the deep learning models are listed below.

2 System Configuration

Hardware and software setup for the research work is explained below with respective diagrams.

2.1 Hardware Configuration

The hardware configuration, Dell Alienware M15 R6 has been used. The specifications are – Operating System - Microsoft Windows 11 Home Single Language, Processor – Intel Core i7, RAM – 16GB, GPU – nvidia 3060 8GB, SSD – 500GB shown in Figure. 1

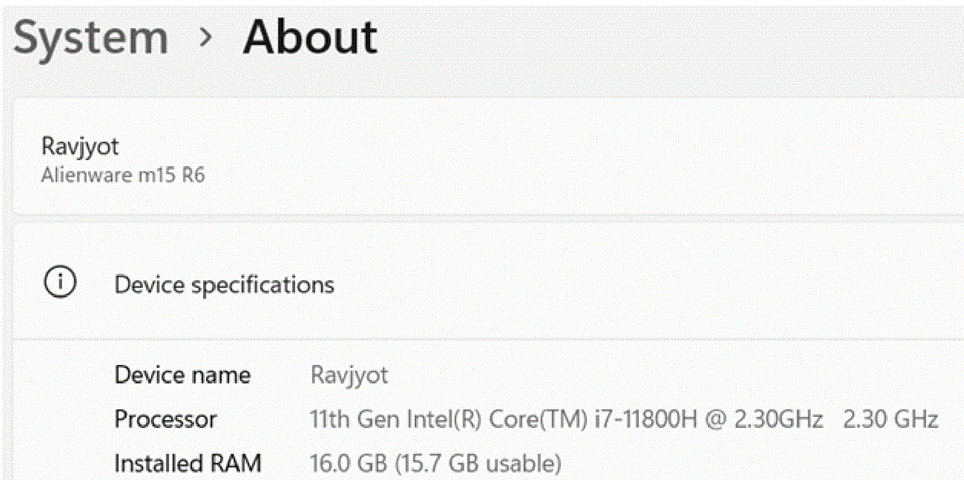


Figure 1: System configuration

2.2 Software Configuration

For software configuration, Python 3.10, Jupyter notebook Figure. 2

- Jupyter Notebook – It is used as the primary GUI for model development purposes.
- Python 3.10 – Python has been used as the main programming language.

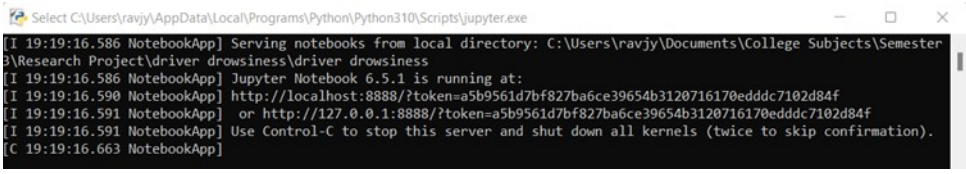


Figure 2: Jupyter notebook

3 Implementation

3.1 Data Source

Dataset can be downloaded from the below. MRL Dataset. [1] Refer Figure. 3

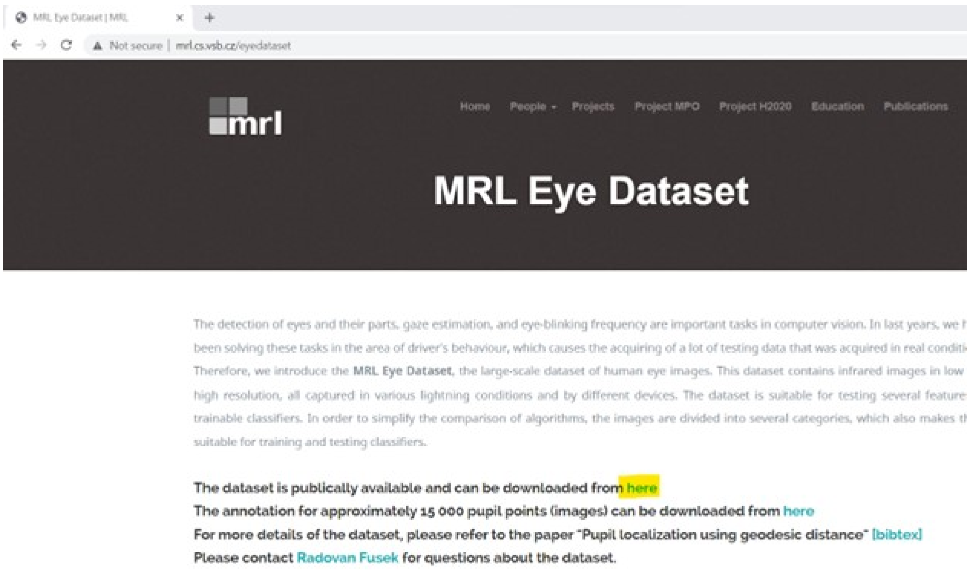


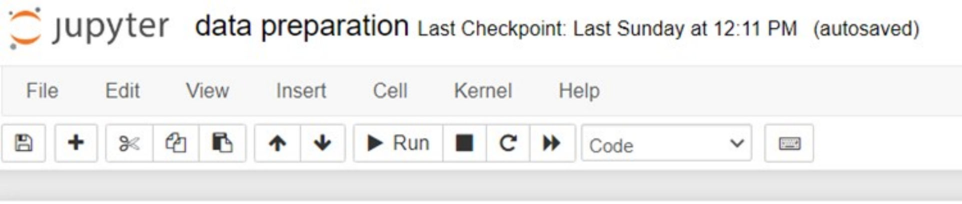
Figure 3: MRL dataset

3.2 Feature Engineering

- Open data preparation code in Jupyter notebook.
- Importing necessary libraries Figure. 4
- Checking the files ending with ".png" format and copying the data into two folders closed eye and open eye. Figure. 5

  https://www.overleaf.com/project/639a3b5a0c912cc351a54e26
- Installing split-folders library Figure. 6
- Splitting the data into train, test, and validation Figure. 7

[1] http://mrl.cs.vsb.cz/eyedataset




```
In [2]: import os
        import shutil
        import glob
        from tqdm import tqdm
```

Figure 4: Importing libraries

```
In [3]: Raw_DIR= r"C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver drowsiness\driver drowsiness\mrlEyes_2018
        for dirpath, dirname, filenames in os.walk(Raw_DIR):
            for i in tqdm([f for f in filenames if f.endswith('.png')]):
                if i.split('_')[4]=='0':
                    shutil.copy(src=dirpath+'/'+i, dst=r"C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver dro

                elif i.split('_')[4]=='1':
                    shutil.copy(src=dirpath+'/'+i, dst=r"C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver dro
```

Figure 5: Checking image format

```
In [4]: !pip install split-folders

        Collecting split-folders
          Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
        Installing collected packages: split-folders
        Successfully installed split-folders-0.5.1
```

Figure 6: Installing library split-folders

```
In [28]: import os
import numpy as np
import shutil
import random
root_dir = base_folder+"input/House_Room_Dataset-5_rooms/" # for requesting directly pics
classes_dir = os.listdir(root_dir)

train_ratio = 0.6
val_ratio  = 0.1

for cls in classes_dir:
    os.makedirs(input_destination +'train_ds/' + cls, exist_ok=True)
    os.makedirs(input_destination +'test_ds/' + cls, exist_ok=True)
    os.makedirs(input_destination +'val_ds/' + cls, exist_ok=True)

    # for each class, let's counts its elements
    src = root_dir + cls
    allFileNames = os.listdir(src)

    # shuffle it and split into train/test/va
    np.random.shuffle(allFileNames)
    train_FileNames, test_FileNames, val_FileNames = np.split(np.array(allFileNames),[int(train_ratio * len(allFileNames)), int(

    # save their initial path
    train_FileNames = [src+'/'+ name  for name in train_FileNames.tolist()]
    test_FileNames  = [src+'/' + name for name in test_FileNames.tolist()]
    val_FileNames   = [src+'/' + name for name in val_FileNames.tolist()]
    print("\n *****************************",
          "\n Total images: ",cls, len(allFileNames),
          '\n Training: ', len(train_FileNames),
          '\n Testing: ', len(test_FileNames),
          '\n Validation: ', len(val_FileNames),
          '\n *****************************')

    # copy files from the initial path to the final folders
    for name in train_FileNames:
        shutil.copy(name, input_destination +'train/' + cls)
    for name in test_FileNames:
        shutil.copy(name, input_destination +'test/' + cls)

# checking everything was fine
paths = ['train/', 'test/']
for p in paths:
    for dir,subdir,files in os.walk(input_destination + p):
    print(dir,' ', p, str(len(files)))
```

Figure 7: Train, test, validation split

3.3 Training the model

- Importing the required library. Figure. 8

```
In [ ]: from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
```

```
In [1]: import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data Augumentation
```

Figure 8: Importing library

- Checking if system GPU is active and available. Figure. 9
- Keras function ImageDataGenerator is used to extract the Train and validation dataset from the prepared dataset folder, with image size defined as - 80 pixels length - 80 pixels width. Figure. 10
- InceptionV3 model's architecture is pre-defined in Keras. By default, InceptionV3 takes input of the shape (299, 299, 3). But it needs to be customized to accommodation the input shape of the images, that is (80, 80, 3) where 3 is the RGB Component. Figure. 11

```
In [5]: from tensorflow.python.client import device_lib
        def get_available_devices():
            local_device_protos = device_lib.list_local_devices()
            return [x.name for x in local_device_protos]
        print(get_available_devices())

In [8]: tf.test.is_gpu_available()

        True

In [4]: with tf.device('/gpu:1'):
            tf.config.list_physical_devices('GPU')
```

Figure 9: System GPU check

```
In [9]: train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0.2,
            zoom_range=0.2,width_shift_range=0.2,
            height_shift_range=0.2, validation_split=0.2)

        train_data= train_datagen.flow_from_directory(r"C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver drow
                                    target_size=(80,80),batch_size=batchsize,class_mode='categorical',subset='training' )

        validation_data= train_datagen.flow_from_directory(r"C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver
                                    target_size=(80,80),batch_size=batchsize,class_mode='categorical', subset='validation')

        Found 64719 images belonging to 2 classes.
        Found 16179 images belonging to 2 classes.

In [10]: test_datagen = ImageDataGenerator(rescale=1./255)

         test_data = test_datagen.flow_from_directory(r'C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver drows
                                    target_size=(80,80),batch_size=batchsize,class_mode='categorical')

         Found 4000 images belonging to 2 classes.
```

Figure 10: Extracting the train and validation dataset

```
In [11]: bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shape=(80,80,3)))
         hmodel = bmodel.output
         hmodel = Flatten()(hmodel)
         hmodel = Dense(64, activation='relu')(hmodel)
         hmodel = Dropout(0.5)(hmodel)
         hmodel = Dense(2,activation= 'softmax')(hmodel)
```

Figure 11: Model's architecture

- Since the pre-trained weights from Imagenet have been used, thus the model needs to be prevented from updating the weights during the training. Thus, for each layer the trainable parameter has been set to False. Figure. 12

```
In [11]: bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shape=(80,80,3)))
         hmodel = bmodel.output
         hmodel = Flatten()(hmodel)
         hmodel = Dense(64, activation='relu')(hmodel)
         hmodel = Dropout(0.5)(hmodel)
         hmodel = Dense(2,activation= 'softmax')(hmodel)

         model = Model(inputs=bmodel.input, outputs= hmodel)
         for layer in bmodel.layers:
             layer.trainable = False
```

Figure 12: Trainable parameter

- Summarizing the model Figure. 13

```
In [13]: model.summary()

Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 80, 80, 3)]  0           []

 conv2d (Conv2D)                (None, 39, 39, 32)   864         ['input_1[0][0]']

 batch_normalization (BatchNorm  (None, 39, 39, 32)  96          ['conv2d[0][0]']
 alization)

 activation (Activation)        (None, 39, 39, 32)   0           ['batch_normalization[0][0]']

 conv2d_1 (Conv2D)              (None, 37, 37, 32)   9216        ['activation[0][0]']

 batch_normalization_1 (BatchNo  (None, 37, 37, 32)  96          ['conv2d_1[0][0]']
 rmalization)

 activation_1 (Activation)      (None, 37, 37, 32)   0           ['batch_normalization_1[0][0]']
```

Figure 13: Model Summary

- Saves the Best model. Figure. 14

```
In [14]: from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping, ReduceLROnPlateau

In [15]: checkpoint = ModelCheckpoint(r"C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver drowsiness 1311\drive
                                      monitor='val_loss',save_best_only=True,verbose=3)
```

Figure 14: Checkpoint

- It monitors the given parameter (validation loss in this case) for given number of patience level (5 in this case). If the value of validation loss does not improve for the 5 consecutive epochs, then the model training needs to stop. Figure. 15
- This also monitors the given parameter (validation loss in this case) for given number of patience level (3 in this case). If the value of validation loss does not improve for the 3 consecutive epochs, then the model training needs to stop. Figure. 16
- Following parameters were used when the model was compiled: Optimizer was chosen to be Adam, loss was chosen to be categorical crossentropy, evaluation metrics were chosen to be categorical accuracy, precision, and recall. 17
- Model training was then initiated with the epoch number set to 50. 18

```
earlystop = EarlyStopping(monitor = 'val_loss', patience=5, verbose= 3, restore_best_weights=True)
```

Figure 15: Early stopping from Keras

```
learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )

callbacks=[checkpoint,earlystop,learning_rate]
```

Figure 16: ReduceLROnPlateau from Keras

```
In [16]: import scipy

In [ ]: model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['categorical_accuracy', tf.keras.metrics.Precision(),
                                                                                   tf.keras.metrics.Recall()])
```

Figure 17: Model compile

```
model.fit(train_data,steps_per_epoch=train_data.samples//batchsize,
                    validation_data=validation_data,
                    validation_steps=validation_data.samples//batchsize,
                    callbacks=callbacks,
                     epochs=50)
```

Figure 18: Model training

4 Evaluation

- During the training, the loss as well as validation loss are coming down with each epoch whereas the categorical accuracy of train and validation datasets are increasing with each epoch. Refer Figure. 19

```
def plot(model):
    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15,4))
    axes[0].plot(model.history.history['loss'])
    axes[0].plot(model.history.history['val_loss'])
    axes[0].legend(['loss','val_loss'])



    axes[1].plot(model.history.history['categorical_accuracy'])
    axes[1].plot(model.history.history['val_categorical_accuracy'])
    axes[1].legend(['categorical_accuracy','val_categorical_accuracy'])
```

Figure 19: Evaluation

- Plot of Train and validation losses over 50 epochs- 20

```
import matplotlib.pyplot as plt
```

```
plot(model)
```

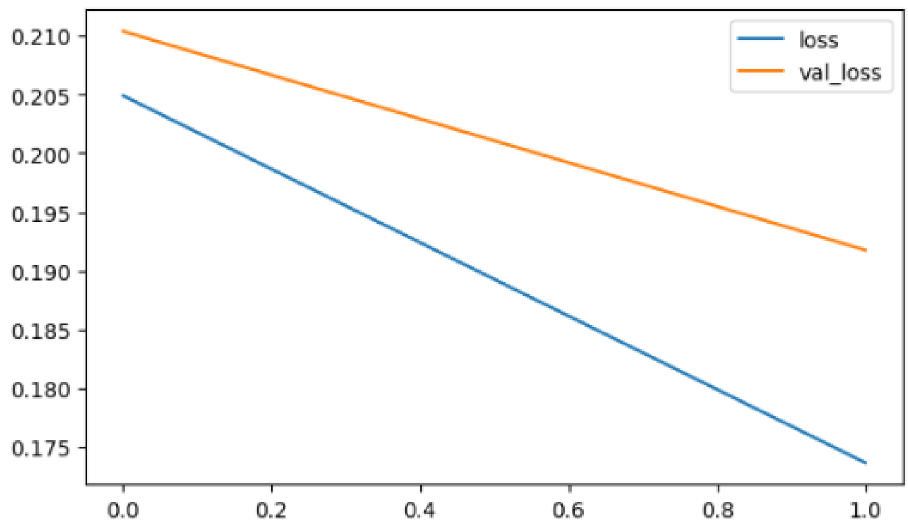


Figure 20: Plot of Train and validation losses over 50 epochs

- Plot of Train and validation categorical accuracies over 50 epochs - 21
- Code for calculating accuracy and loss for train, validation and test datasets - 22

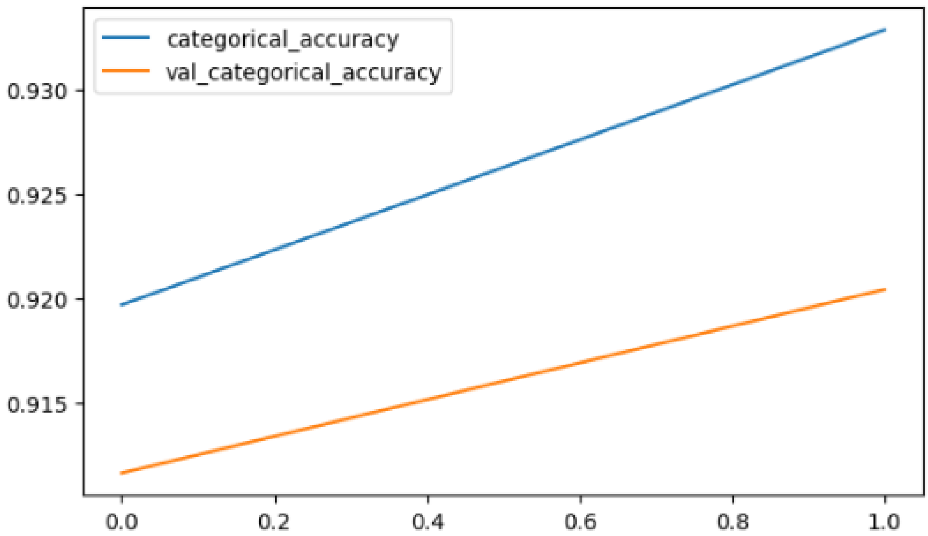


Figure 21: Plot of Train and validation categorical accuracies over 50 epochs

```
In [ ]: acc_tr, loss_tr = model.evaluate(train_data)
        print(acc_tr)
        print(loss_tr)

In [ ]: acc_vr, loss_vr = model.evaluate(validation_data)
        print(acc_vr)
        print(loss_vr)

In [ ]: acc_test, loss_test = model.evaluate(test_data)
        print(acc_tr)
        print(loss_tr)
```

Figure 22: Code for calculation of categorical accuracies and losses

5 Running the model

- Importing the libraries Refer Figure. 23

jupyter Main Production Last Checkpoint: 12/04/2022 (unsaved changes)

File Edit View Insert Cell Kernel Help

```
In [1]: import cv2
        import tensorflow as tf
        from tensorflow import keras
        from tensorflow.keras.models import load_model
        import numpy as np
        from pygame import mixer

pygame 2.1.2 (SDL 2.0.18, Python 3.10.8)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

Figure 23: Importing libraries

- Importing Haar cascade files Refer Figure. 24

```
In [7]: face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
        model = load_model(r"C:\Users\ravjy\Documents\College Subjects\Semester 3\Research Project\driver drowsiness\driver drowsiness\mo
```

Figure 24: Haar cascade files

- Code for early stop. 25

```
earlystop = EarlyStopping(monitor = 'val_loss', patience=5, verbose= 3, restore_best_weights=True)
```

Figure 25: early stop

- Code for model compile and fit. 26
- Code for live camera feed. 27

```
In [16]: import scipy

In [ ]: model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['categorical_accuracy', tf.keras.metrics.Precision(),
                                                                                tf.keras.metrics.Recall()])

        model.fit(train_data,steps_per_epoch=train_data.samples//batchsize,
                            validation_data=validation_data,
                            validation_steps=validation_data.samples//batchsize,
                            callbacks=callbacks,
                             epochs=50)
```

Figure 26: Model compile and fit

```
import winsound
frequency = 2500  # Set frequency to 2500
duration = 1500  # Set duration to 1500 ms == 1.5 sec
import numpy as np
import cv2
path = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
cap = cv2.VideoCapture(0)
#check if webcam is opened correctly
if not cap.isOpened():
    cap = cv2.VideoCapture(1)
cap.set(cv2.CAP_PROP_FPS, 15)
counter = 0
while True:
    ret,frame = cap.read()
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    eyes = eye_cascade.detectMultiScale(gray, 1.1, 4)
    for x,y,w,h in eyes:
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 255, 0), 2)
        eyess = eye_cascade.detectMultiScale(roi_gray)
        if len(eyess) == 0:
            print("Eyes are not detected")
        else:
            for (ex, ey, ew, eh) in eyess:
                eyes_roi = roi_color[ey: ey+eh, ex: ex+ew]
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    if(faceCascade.empty()==False):
        print("detected")
    faces = faceCascade.detectMultiScale(gray, 1.1, 4)
    # Draw a rectangle around eyes
    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    font = cv2.FONT_HERSHEY_SIMPLEX
    final_image = cv2.resize(eyes_roi, (80,80))
    final_image = np.expand_dims(final_image, axis=0)
    final_image = final_image/255.0
    Predictions = model.predict(final_image)
    print(Predictions)
```

Figure 27: Live camera feed