A comprehensive comparison analysis of scholarly investigations on Human Iris detection on deep neural network

MSc Research Project
Data Analytics

Debmalya Deb
Student ID: x21242101

School of Computing
National College of Ireland

Supervisor: Aaloka Anant

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Debmalya Deb

Student ID: x21242101

Programme: MSc Data Analytics **Year:** 2023

Module: MSc Research Project

Lecturer: Aaloka Anant

Submission Due Date: 14/12/2023

Project Title: A comprehensive comparison analysis of scholarly investigations on Human Iris detection on deep neural network

Word Count:725 **Page Count:**10

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: Debmalya Deb

Date: 14/12/2023

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):	

A comprehensive comparison analysis of scholarly investigations on Human Iris detection on deep neural network

Debmalya Deb
x21242101

1 Introduction

This document aims to provide the entire configuration and setup details of this Iris Detection thesis project, including the model building with ResNet152V2 and CNN model. This manual will be followed to set up the code and detection model. This code was performed in the 3 different IDEs due to some computation resources and technical constrains. The three IDEs are Jupyter Notebook, Google Colab Pro and Pycharm.

2 Hardware and Software

This project was entirely performed on a MacBook M1 laptop and Figure 1 shows the hardware configuration of the laptop.

Figure 1: MacBook configuration setup

Now Figure 2 represents the Jupyter Notebook version. In Jupyter Notebook the Dataset creation, Data Pre-processing, and Data augmentation have been performed.

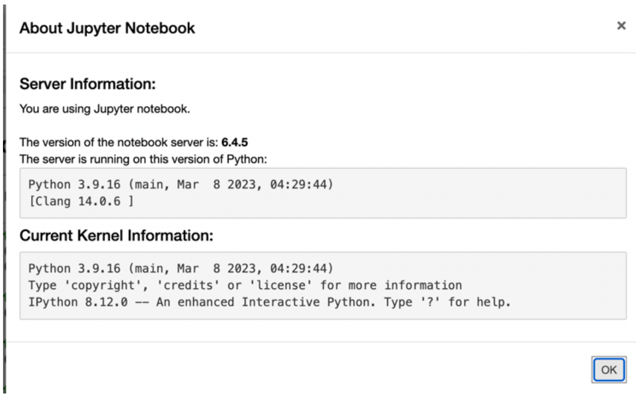

Figure 2: Jupyter notebook version

Figure 3 presents the Google Colab Pro Configuration details where the actual model building has been done and both of the models have been trained in Google Colab Pro.

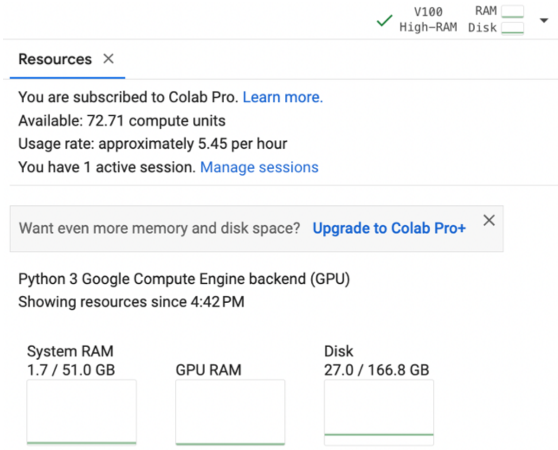

Figure 3: Google Colab Pro Version

Figure 4 represents the configuration details of the PyCharm. The models have been deployed in the PyCharm and here only the real-time tracking trials performed although it's on the future scope.

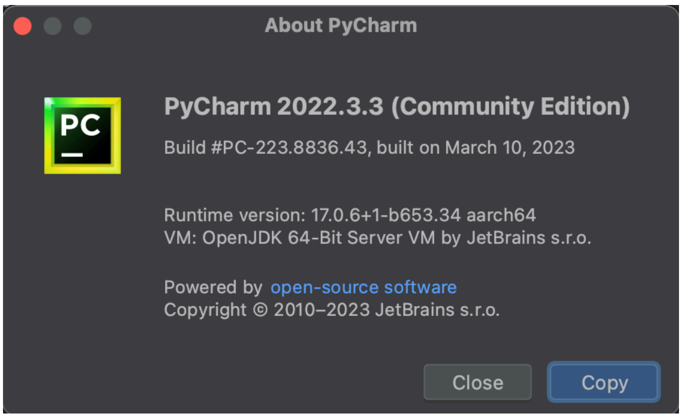

Figure 4: PyCharm version

3 Dataset Collection

Short videos were collected from https://www.kaggle.com/code/mmmarchetti/deep-fake-chalenge/input based on different iris movement scenarios. Figure 5 shows the collected videos in the local directory.

- v1.mp4
- v2.mp4
- v3.mp4
- v4.mp4
- v5.mp4
- v6.mp4
- v7.mp4
- v8.mp4
- v9.mp4
- v10.mp4
- v11.mp4
- v12.mp4
- v13.mp4
- v14.mp4
- v15.mp4
- v16.mp4
- v17.mp4
- v18.mp4
- v19.mp4

Figure 5: Selected videos from Kaggle

4 Implementation

4.1 Dataset Preparation

Figure 6 shows how images were taken from the videos in the Jupyter Notebook followed by Figure 7 like how the author's images with different iris movements were captured using built-in webcam.

```python
import os
import time
import uuid
import cv2
```

```python
VIDEO_PATH = os.path.join('thesis image','video','v18.mp4')
IMAGES_PATH = os.path.join('thesis image','images')
number_images = 5
```

```python
# Specify the desired width and height for the saved images
image_width = 450
image_height = 450
```

```python
# Create a VideoCapture object with the video file
cap = cv2.VideoCapture(VIDEO_PATH)

# Check if the video file is opened successfully
if not cap.isOpened():
    print('Error: Could not open the video file.')
    exit()

total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
frame_skip = total_frames // number_images

for imgnum in range(number_images):
    print('Collecting image {}'.format(imgnum))

    frame_number = imgnum * frame_skip
    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)

    ret, frame = cap.read()

    if not ret:
        print('End of video reached.')
        break

    # Resize the captured frame to the desired width and height
    frame = cv2.resize(frame, (image_width, image_height))

    imgname = os.path.join(IMAGES_PATH, f'{str(uuid.uuid1())}.jpg')
    cv2.imwrite(imgname, frame)
    cv2.imshow('frame', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

```
Collecting image 0
Collecting image 1
Collecting image 2
Collecting image 3
Collecting image 4
```

Figure 6: - Image captured from videos.

```python
import os #it provides fuctions to insteract with the operating system
import time # it helps to provide the time which will be used to set delay between the image capture
import uuid # helps to create unique images guves unique id while image creation
import cv2 # helps to create video frame
```

```python
IMAGES_PATH = os.path.join('thesis image','images') # defines path where the captured images will going to be sa
number_images = 15 # number of images capture
desired_width = 450
desired_height = 450
```

```python
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print('Error: Could not open camera.')
    exit()

for imgnum in range(number_images):
    print('Collecting image {}'.format(imgnum))

    ret, frame = cap.read()

    if not ret:
        print('Failed to capture a frame.')
        break

    imgname = os.path.join(IMAGES_PATH, f'{str(uuid.uuid1())}.jpg')

    # Resize the captured frame to the desired dimensions (450x450)
    frame = cv2.resize(frame, (desired_width, desired_height))

    cv2.imwrite(imgname, frame)
    cv2.imshow('frame', frame)
    time.sleep(0.5)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

```
Collecting image 0
Collecting image 1
Collecting image 2
Collecting image 3
Collecting image 4
Collecting image 5
Collecting image 6
Collecting image 7
Collecting image 8
Collecting image 9
Collecting image 10
Collecting image 11
Collecting image 12
Collecting image 13
Collecting image 14
```

Figure 7: Author’s images captured using the webcam.

4.2 Data Labelling

Figure 8 represents how to open the LabelMe window for data annotation.

```python
!labelme # initiate the labelme window
```

Figure 8:- Data Annotation using labelMe

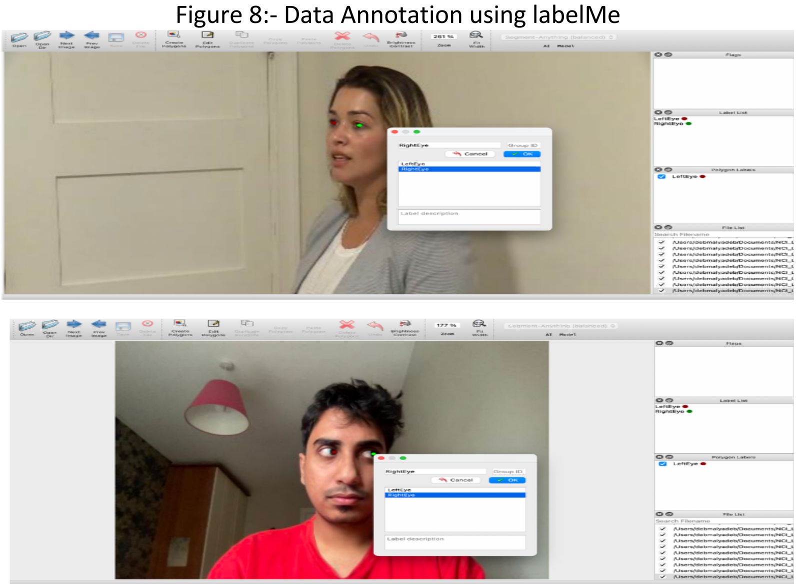

Figure 9: Image Annotation

Figure 9 shows the annotation procedure. Here the paper first chose the directory where all the unique labelled images were stored and with that, an output directory was also selected where the JSON files containing all labelled pieces of information would get stored. The study had to select the class name and the key point annotation colour. For “RightEye” the colour was green and for “LeftEye” the colour was red.

4.3 Data scaling

Figure 10 shows how the images were cropped into 450x450.

```
import tensorflow as tf
import cv2
import json
import numpy as np
from matplotlib import pyplot as plt
```

```
2023-10-17 15:47:12.897155: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimize
d with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical op
erations:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

```
# # Avoid OOM errors by setting GPU Memory Consumption Growth
# gpus = tf.config.experimental.list_physical_devices('GPU')
# for gpu in gpus:
#     tf.config.experimental.set_memory_growth(gpu, True)
```

```
images = tf.data.Dataset.list_files('thesis image/images/*.jpg')
```

```
images.as_numpy_iterator().next()
```

```
b'thesis image/images/e3b0be3c-6cf5-11ee-bf8b-e251dab298e8.jpg'
```

```
# def load_image(x):
#     byte_img = tf.io.read_file(x)
#     img = tf.io.decode_jpeg(byte_img)
#     return img
```

```
def load_and_resize_image(x):
    byte_img = tf.io.read_file(x)
    img = tf.io.decode_jpeg(byte_img)
    img = tf.image.resize(img, [450, 450])  # Resize to the desired shape
    return img
```

```
images = images.map(load_and_resize_image)
```

Figure 10:- Cropped the images into 450x450

The cropped images were plotted to check their sizes as shown in figure 11.

```
type(images)
```

```
tensorflow.python.data.ops.dataset_ops.MapDataset
```

```
image_generator = images.batch(4).as_numpy_iterator()
```

```
plot_images = image_generator.next()
```

```
# # Plot the images with circles
# fig, ax = plt.subplots(ncols=4, figsize=(20, 20))
# for idx, images in enumerate(plot_images):
#     ax[idx].imshow(images)

# plt.show()

# Normalize the images to the [0, 1] range
plot_images = plot_images / 255.0

# Plot the images with circles
fig, ax = plt.subplots(ncols=4, figsize=(20, 20))
for idx, images in enumerate(plot_images):
    ax[idx].imshow(images)

plt.show()
```

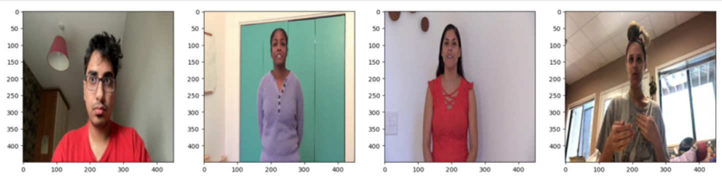

Figure 11:- Scale visualisation of the snaps.

The images were split and moved into the train, test and validation directory manually, 75% were sent to train and test and the validation dataset got 15% each of the entire dataset. After spiting the Figure 12 explains how the annotations were moved with respect to their located folders.

```
import os
```

```
for folder in ['train','test','val']:
    for file in os.listdir(os.path.join('thesis image', folder, 'images')):

        filename = file.split('.')[0]+'.json'
        existing_filepath = os.path.join('thesis image','labels', filename)
        if os.path.exists(existing_filepath):
            new_filepath = os.path.join('thesis image',folder,'labels',filename)
            os.replace(existing_filepath, new_filepath)
```

Figure 12:- Moving the annotations

4.4 Data Augmentation

The images and their labels were loaded into the respective functions as per Figure 13.

augmentation code

```
In [1]: import albumentations as alb
        import cv2
        import os
        import json
        import numpy as np

In [2]: # import os
        # print(os.getcwd())

In [3]: img = cv2.imread(os.path.join('thesis image', 'train', 'images', '2f251ebc-6cf6-11ee-bcbb-e251dab298e8.jpg'))

In [4]: img.shape # we can see the image sizes are random so that we are gonna crop that down in the below code
Out[4]: (450, 450, 3)

In [5]: with open(os.path.join('thesis image', 'train', 'labels', '2f251ebc-6cf6-11ee-bcbb-e251dab298e8.json'), 'r') as
            label = json.load(f)

In [6]: label
Out[6]: {'version': '5.3.1',
         'flags': {},
         'shapes': [{'label': 'LeftEye',
           'points': [[196.4689265536723, 139.40677966101694]],
           'group_id': None,
           'description': '',
           'shape_type': 'point',
           'flags': {}},
          {'label': 'RightEye',
           'points': [[218.50282485875704, 133.19209039548022]],
           'group_id': None,
           'description': '',
           'shape_type': 'point',
           'flags': {}}],
         'imagePath': '../images/2f251ebc-6cf6-11ee-bcbb-e251dab298e8.jpg',
```


'imageData': '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAx
NDQ0Hyc5PTgyPC4zNDL/2wBDAQgJCQwLDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjL/wAA
RCAHCAcIDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhBy
JxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl

Figure 13:- Images and Labels are getting stored

All the coordinates had checked whether they were coming properly or not, as described in Figure 14.

```
In [14]: label['shapes']
Out[14]: [{'label': 'LeftEye',
           'points': [[196.4689265536723, 139.40677966101694]],
           'group_id': None,
           'description': '',
           'shape_type': 'point',
           'flags': {}},
          {'label': 'RightEye',
           'points': [[218.50282485875704, 133.19209039548022]],
           'group_id': None,
           'description': '',
           'shape_type': 'point',
           'flags': {}}]

In [ ]:
```

check the coordinates

```
In [15]: coords = [0,0,0,0]
         coords[0] = label['shapes'][0]['points'][0][0]
         coords[1] = label['shapes'][0]['points'][0][1]
         coords[2] = label['shapes'][1]['points'][0][0]
         coords[3] = label['shapes'][1]['points'][0][1]

In [16]: coords
Out[16]: [196.4689265536723, 139.40677966101694, 218.50282485875704, 133.19209039548022]

In [17]: coords = list(np.divide(coords, [640,480,640,480]))

In [18]: coords

Out[18]: [0.306982697740113,
          0.29043079096045193,
          0.34141066384180785,
          0.2774835216572505]
```

Figure 14:- Iris coordinates

Figure 15 and Figure 16 show the python albumentation function and the augmentation pipeline.

```
# https://albumentations.ai/
# we can keep the frame as same as bbox annotation

augmentor = alb.Compose([alb.RandomCrop(width=450, height=450),
                         alb.HorizontalFlip(p=0.5),
                         alb.RandomBrightnessContrast(p=0.2),
                         alb.RandomGamma(p=0.2),
                         alb.RGBShift(p=0.2),
                         alb.VerticalFlip(p=0.5)],
                        keypoint_params=alb.KeypointParams(format='xy', label_fields=['class_labels']))

# xy format is being used for keyPoint annotations https://albumentations.ai/docs/getting_started/keypoints_augm
```

Figure 15:- Python Albumentation code

augmentation pipeline

```
for partition in ['train', 'test', 'val']:
    for image in os.listdir(os.path.join('thesis image', partition, 'images')):
        img = cv2.imread(os.path.join('thesis image', partition, 'images', image))

        classes = [0,0]
        coords = [0,0,0.00001,0.00001]
        label_path = os.path.join('thesis image', partition, 'labels', f'{image.split(".")[0]}.json')
        if os.path.exists(label_path):
            with open(label_path, 'r') as f:
                label = json.load(f)

            if label['shapes'][0]['label']=='LeftEye':
                classes[0] = 1
                coords[0] = np.squeeze(label['shapes'][0]['points'])[0]
                coords[1] = np.squeeze(label['shapes'][0]['points'])[1]

            if label['shapes'][0]['label']=='RightEye':
                classes[1] = 1
                coords[2] = np.squeeze(label['shapes'][0]['points'])[0]
                coords[3] = np.squeeze(label['shapes'][0]['points'])[1]

            if len(label['shapes']) > 1:
                if label['shapes'][1]['label'] =='LeftEye':
                    classes[0] = 1
                    coords[0] = np.squeeze(label['shapes'][1]['points'])[0]
                    coords[1] = np.squeeze(label['shapes'][1]['points'])[1]

                if label['shapes'][1]['label'] =='RightEye':
                    classes[1] = 1
                    coords[2] = np.squeeze(label['shapes'][1]['points'])[0]
                    coords[3] = np.squeeze(label['shapes'][1]['points'])[1]

            np.divide(coords, [640,480,640,480])

        try:
            for x in range(120):
                keypoints = [(coords[:2]), (coords[2:])]
                augmented = augmentor(image=img, keypoints=keypoints, class_labels=['LeftEye','RightEye'])
                cv2.imwrite(os.path.join('aug_data', partition, 'images', f'{image.split(".")[0]}.{x}.jpg'), aug

                annotation = {}
                annotation['image'] = image
                annotation['class'] = [0,0]
                annotation['keypoints'] = [0,0,0,0]

                if os.path.exists(label_path):
                    if len(augmented['keypoints']) > 0:
                        for idx, cl in enumerate(augmented['class_labels']):
                            if cl == 'LeftEye':
                                annotation['class'][0] = 1
                                annotation['keypoints'][0] = augmented['keypoints'][idx][0]
                                annotation['keypoints'][1] = augmented['keypoints'][idx][1]
                            if cl == 'RightEye':
                                annotation['class'][1] = 1
                                annotation['keypoints'][2] = augmented['keypoints'][idx][0]
                                annotation['keypoints'][3] = augmented['keypoints'][idx][1]

                annotation['keypoints'] = list(np.divide(annotation['keypoints'], [450,450,450,450]))


                with open(os.path.join('aug_data', partition, 'labels', f'{image.split(".")[0]}.{x}.json'), 'w')
                    json.dump(annotation, f)

        except Exception as e:
            print(e)
```

```
image must be numpy array type
image must be numpy array type
image must be numpy array type
```

Figure 16:- Data Augmentation Pipeline

5 Model Building

The augmented dataset was mounted on the google colab and the essential Python libraries were imported as shown in Figure 17.

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

```
Mounted at /content/drive
```

```python
import os
import cv2
import tensorflow as tf
import cv2
import json
import numpy as np
from matplotlib import pyplot as plt
```

Figure 17:- Dataset mount and important python libraries

Before modeling all the augmented images and their annotations were gathered. Figure 18 shows all together 8280 images were there in the training dataset, 1800 were in the test and 1800 were in the validation dataset.

```python
len(train_images)

```

```
8280
```

```python
len(test_images)

```

```
1800
```

```python
len(val_images)
```

```
1800
```

Figure 18:- Total number of augmented data

Finally, the images and annotations were zipped for train, validation and testing. Figure 19 visualises the key point annotation with the images.

```python
fig, ax = plt.subplots(ncols=4, figsize=(20,20)) # 4 different colomns to visualise the images and figsize is for how big I want
for idx in range(4): # looping 4 different images
    sample_image = res[0][idx] # access our key which is in index 0
    sample_coords = res[1][0][idx] # and we are going to the corresponding coordinates
    # open cv to draw the points with the perfect coordinates in 255 pixles because we scaled the images before in 255 pixles and we divided it before and that
    # why we are multipling here
    cv2.circle(sample_image, tuple(np.multiply(sample_coords[:2], [250,250]).astype(int)), 2, (255,0,0), -1)
    cv2.circle(sample_image, tuple(np.multiply(sample_coords[2:], [250,250]).astype(int)), 2, (0,255,0), -1)

    ax[idx].imshow(sample_image)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

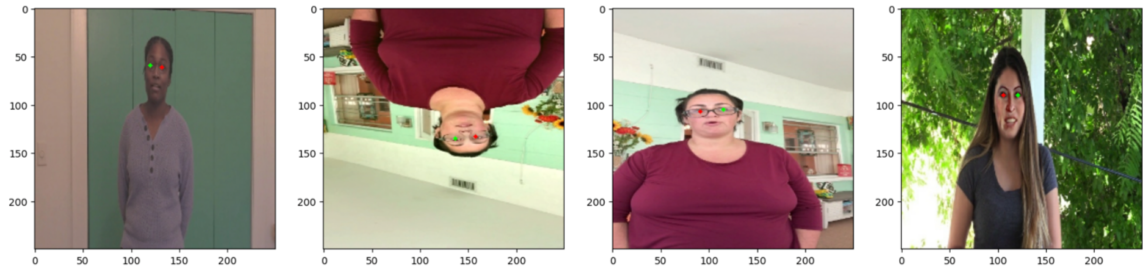

Figure 19:- Zipped images with annotations

5.1 ResNet Model

Figure 20 represents the ResNet model building.

```python
from tensorflow.keras.models import Sequential # sequestinal neural network
from tensorflow.keras.layers import Input, Conv2D, Reshape, Dropout # input defines the shape, convolution neural network, to reshape, dropout for regularisation
from tensorflow.keras.applications import ResNet152V2 # trasnfer learning module/ pre-existing neural network
```

```python
# input shape has to 250,250,3 and here I am using padding and relu activation fucntion throughout
model = Sequential([
    Input(shape=(250,250,3)),
    ResNet152V2(include_top=False, input_shape=(250,250,3)),
    Conv2D(512, 3, padding='same', activation='relu'),
    Conv2D(512, 3, padding='same', activation='relu'),
    Conv2D(256, 3, 2, padding='same', activation='relu'), # 2 represents the stride
    Conv2D(256, 2, 2, activation='relu'),
    Dropout(0.05),
    Conv2D(4, 2, 2),
    Reshape((4,))
])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet152v2_weights_tf_dim_ordering_tf_kernels_notop.h5
234545216/234545216 [==============================] - 1s 0us/step
```

Figure 20:- The ResNet model

5.2 CNN Model

Figure 21 represents the CNN model building.

```python
from tensorflow.keras.models import Sequential # sequestinal neural network
from tensorflow.keras.layers import Input, Conv2D, Reshape, Dropout # input defines the shape, convolution neural network, to reshape, dropout for regularisation
from tensorflow.keras.applications import MobileNetV2 # trasnfer learning module/ pre-existing neural network
```

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv2D, Activation, MaxPooling2D, Flatten, Dense, Dropout, Reshape

model = Sequential([
    Input(shape=(250, 250, 3)),
    Conv2D(64, 3, padding='same', activation='relu'),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(128, 3, padding='same', activation='relu'),
    Conv2D(128, 3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(256, 3, padding='same', activation='relu'),
    Conv2D(256, 3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),

    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(4),
    Reshape((4,))
])

```

Figure 21:- The CNN model

5.3 Deployment

Figure 22 shows the code to load the model in .h5 format.

Save the model for real time performance

```
[ ] from tensorflow.keras.models import load_model # load model module from keras tuner
```

```
[ ] model.save('/content/drive/MyDrive/iristrackerresnet.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is consi
  saving_api.save_model(
```

```
[ ] model = load_model('/content/drive/MyDrive/iristrackerresnet.h5')
```

Figure 22:- Saving the model for deployment

Finally, the future scope of this thesis was real-time tracking. Figure 23 shows the code of real-time tracking. This code for real-time tracking was written in PyCharm.

```
from tensorflow.keras.models import load_model
import cv2
import numpy as np

model = load_model('iristrackerresnet.h5')

cap = cv2.VideoCapture(0)
while cap.isOpened():
    _, frame = cap.read()

    frame = frame[50:500, 50:500, :]
    rgb_img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    resized = cv2.resize(rgb_img, (250, 250))

    yhat = model.predict(np.expand_dims(resized / 255, 0))
    sample_coords = yhat[0, :4]

    cv2.circle(frame, tuple(np.multiply(sample_coords[:2], [450, 450]).astype(int)), 2, (255, 0, 0), -1)
    cv2.circle(frame, tuple(np.multiply(sample_coords[2:], [450, 450]).astype(int)), 2, (0, 255, 0), -1)

    cv2.imshow('EyeTrack', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Figure 23:- Model deployment code