MCTA 4363 Deep Learning: Quiz 2 (30 marks)

Name: Sofeatul Aqida binti Mohd Yusof Matric No: 2029704

Fig. 1

Fig. 1 shows a photo of participants in a workshop. As a deep learning engineer, you are tasked with developing an object detection system to detect the faces and count the number of male and female participants in the photo using YOLO v8 and OpenCV.

Marks Distribution:

a) Dataset collection (10 marks)
b) Model selection and training (10 marks)
c) Code for model inference and counting (10 marks)

Instructions for Submission:

1. Submit a single PDF file through the provided link in Microsoft Teams.
2. In the PDF, include:
   - the code snippet,
   - output of the code and
   - the GitHub link containing the code, model and dataset used for training.

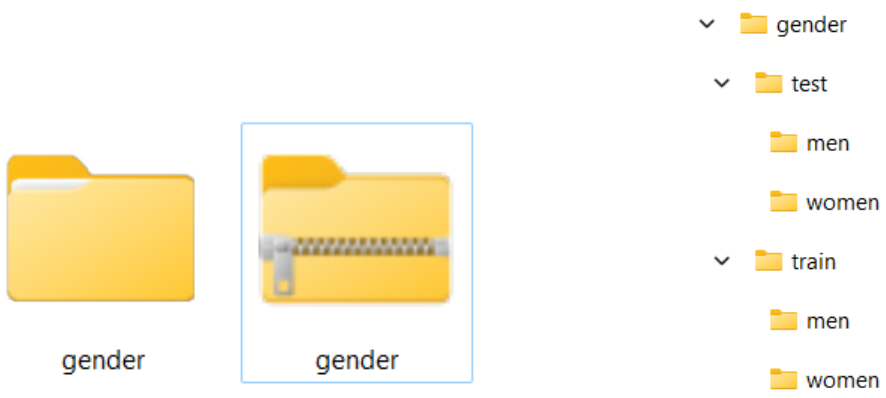
a) Dataset Collection

The dataset collection process involves gathering and preparing the dataset required for training and testing the object detection model.

I have collected and downloaded the dataset from,

1) https://www.kaggle.com/datasets/mostafaebrahiem/women-faces-with-hijabscientific-use-only
2) https://www.kaggle.com/datasets/saadpd/menwomen-classification
3) and some random Google Images for the women who are wearing niqab.

The dataset was then organized into a structured directory, with 800 images designated for training and 220 images for testing. These images were further categorized into 'men' and 'women' folders within both the training and testing directories for a clear and systematic organization.



(a) (b)

Figure 1: (a) Folder gender with gender.zip (b) Dataset structured format

b) Model selection and training

For the model selection and training phase, I use Google Collab to handle custom datasets and perform data augmentation. The process began with defining the dataset paths, specifying the number of classes, and setting class names, ensuring the model could distinguish between male and female faces accurately. The training script included essential data augmentation techniques to enhance model robustness by exposing it to various transformations of the input images. Hyperparameters were carefully tuned, with the learning

rate, batch size, and number of epochs adjusted to optimize performance. Throughout the training, accuracy and loss metrics were monitored and visualized using graphs, providing real-time feedback on the model's progress. After multiple epochs of training and validation, the model achieving the best performance was saved as `good_model(3).pt`. This model demonstrated superior accuracy and generalization, making it well-suited for the subsequent object detection and counting tasks. The use of data augmentation and hyperparameter tuning was instrumental in achieving a high-performing and reliable model.

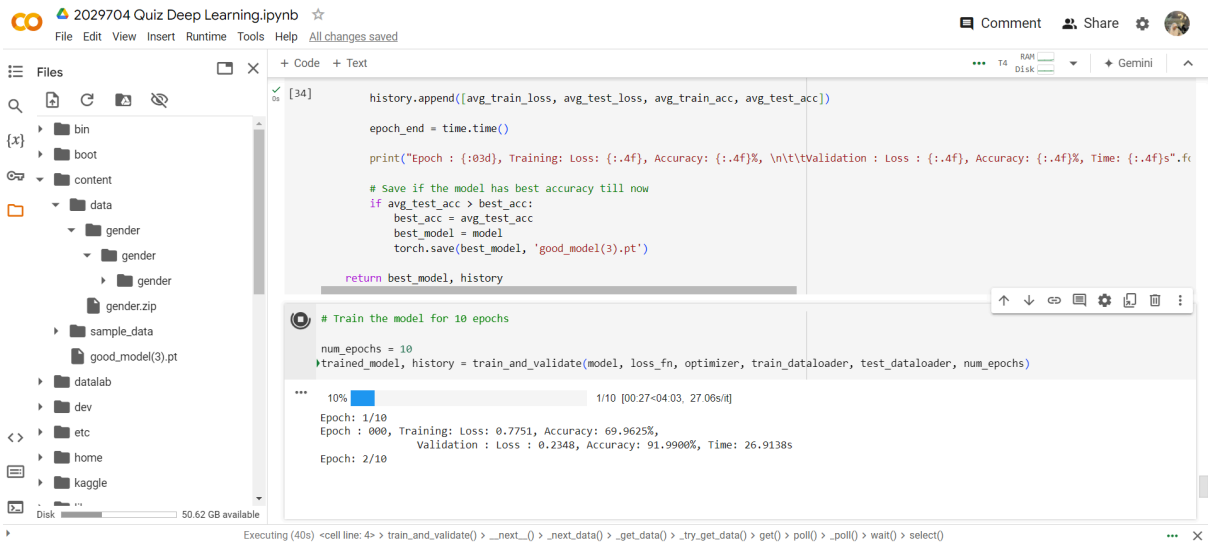


Figure 2: Google Collab Model Selection and Training

```
[7] import os
    def walk_through_dir(dir_path):

      for dirpath, dirnames, filenames in os.walk(dir_path):
        print(f"There are {len(dirnames)} directories and {len(filenames)} images in '{dirpath}'.")

    walk_through_dir(image_path)
```

```
There are 1 directories and 0 images in 'data/gender/gender'.
There are 2 directories and 0 images in 'data/gender/gender/gender'.
There are 2 directories and 0 images in 'data/gender/gender/gender/test'.
There are 0 directories and 110 images in 'data/gender/gender/gender/test/women'.
There are 0 directories and 110 images in 'data/gender/gender/gender/test/men'.
There are 2 directories and 0 images in 'data/gender/gender/gender/train'.
There are 0 directories and 400 images in 'data/gender/gender/gender/train/women'.
There are 0 directories and 400 images in 'data/gender/gender/gender/train/men'.
```

```
[12] # Get class names as a list
     class_names = train_data.classes
     class_names
```

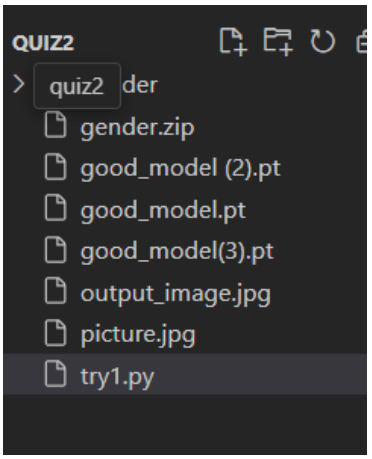
```
['men', 'women']
```

Figure 3: dataset paths, specifying the number of classes, and setting class names.

```
[35]  100%|██████████| 10/10 [04:26<00:00, 26.34s/it]
Epoch: 1/10
Epoch : 000, Training: Loss: 0.7751, Accuracy: 69.9625%,
                Validation : Loss : 0.2348, Accuracy: 91.9900%, Time: 26.9138s
Epoch: 2/10
Epoch : 001, Training: Loss: 0.4908, Accuracy: 81.6020%,
                Validation : Loss : 0.0780, Accuracy: 97.2466%, Time: 26.5753s
Epoch: 3/10
Epoch : 002, Training: Loss: 0.3449, Accuracy: 88.3605%,
                Validation : Loss : 0.0518, Accuracy: 98.1227%, Time: 26.3060s
Epoch: 4/10
Epoch : 003, Training: Loss: 0.3906, Accuracy: 84.9812%,
                Validation : Loss : 0.1024, Accuracy: 97.3717%, Time: 26.7822s
Epoch: 5/10
Epoch : 004, Training: Loss: 0.2494, Accuracy: 90.7384%,
                Validation : Loss : 0.0259, Accuracy: 99.1239%, Time: 26.8969s
Epoch: 6/10
Epoch : 005, Training: Loss: 0.2720, Accuracy: 91.2390%,
                Validation : Loss : 0.0166, Accuracy: 99.6245%, Time: 27.5389s
Epoch: 7/10
Epoch : 006, Training: Loss: 0.3025, Accuracy: 89.9875%,
                Validation : Loss : 0.0151, Accuracy: 99.6245%, Time: 26.2242s
Epoch: 8/10
Epoch : 007, Training: Loss: 0.3368, Accuracy: 87.7347%,
                Validation : Loss : 0.0095, Accuracy: 99.6245%, Time: 25.8860s
Epoch: 9/10
Epoch : 008, Training: Loss: 0.1531, Accuracy: 93.6170%,
                Validation : Loss : 0.0107, Accuracy: 99.7497%, Time: 25.9523s
Epoch: 10/10
Epoch : 009, Training: Loss: 0.3522, Accuracy: 87.7347%,
                Validation : Loss : 0.0088, Accuracy: 99.7497%, Time: 26.2693s
```

Figure 4: Train the model for 10 epochs and get good_model(3).pt

I have trained the model multiple times and got three models named good_model.pt, good_model (2).pt and lastly good_model(3).pt. The output image below is from good_model (2).pt, and this is the best output so far. Below shows the output images differences compared to the three models:



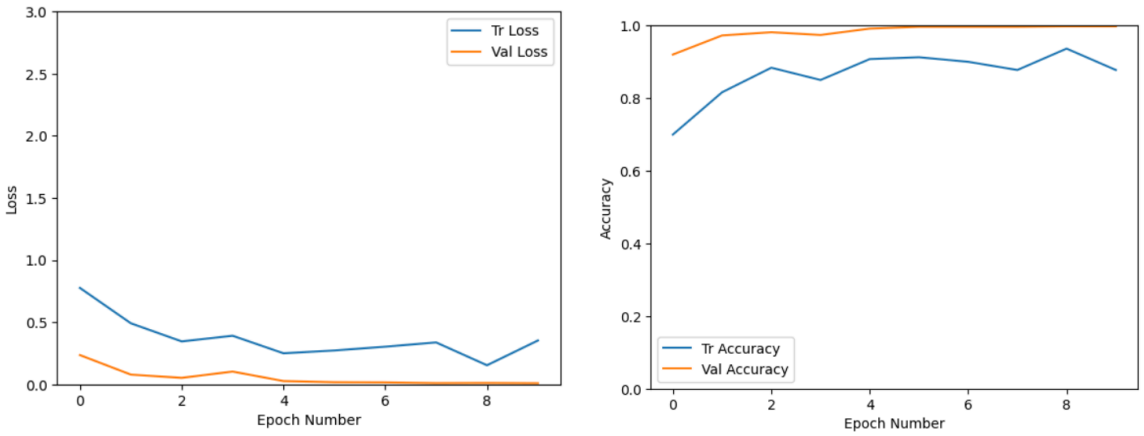


Figure 5: Accuracy and Loss metrics were monitored and visualized using graphs

Disclaimer: The Google Collab that I sent in Github and this report was from good_model(3).pt. There is the loss of code for previously trained models for good_model.pt and good_model(2).pt.

c) Code for model inference and counting

Visual Studio code try1.py inference and counting:

```python
import torch
import cv2
import numpy as np
from PIL import Image
from torchvision import transforms

# Check if CUDA is available and set device accordingly
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load the pretrained PyTorch model
model = torch.load('good_model (2).pt', map_location=device)
model.eval()
model.to(device)

# Define the preprocessing transformations
preprocess = transforms.Compose([
    transforms.Resize((224, 224)),  # Adjust according to your model's input size
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

# Load the face detector (Haar Cascade)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Load the image
image_path = 'picture.jpg'
image = cv2.imread(image_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```python
# Detect faces in the image
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

num_males = 0
num_females = 0

for (x, y, w, h) in faces:
    # Extract the face region
    face = image[y:y+h, x:x+w]
    face_rgb = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)

    # Convert the face to PIL Image
    face_pil = Image.fromarray(face_rgb)

    # Preprocess the face
    face_tensor = preprocess(face_pil)
    face_tensor = face_tensor.unsqueeze(0)  # Add batch dimension
    face_tensor = face_tensor.to(device)  # Move tensor to the correct device

    # Perform inference
    with torch.no_grad():
        outputs = model(face_tensor)
        prediction = outputs.argmax(dim=1).item()  # Get the predicted class (0 for male, 1 for female)

    # Update counters and set colors
    if prediction == 0:
        label = "Female"
        num_females += 1
        box_color = (255, 0, 255)  # Pink for female
        text_color = (255, 0, 255)  # Pink text for female
    else:
        label = "Male"
        num_males += 1
        box_color = (0, 0, 255)  # Red for male
        text_color = (0, 0, 255)  # Red text for male
```

```
    # Draw the bounding box and label on the image
    cv2.rectangle(image, (x, y), (x+w, y+h), box_color, 2)
    cv2.putText(image, label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, text_color, 2)

# Display the counter on the top left corner
cv2.putText(image, f'Males: {num_males}  Females: {num_females}', (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

# Save the output image
output_image_path = 'output_image.jpg'
cv2.imwrite(output_image_path, image)

# Resize the window to fit the image
cv2.namedWindow('Image', cv2.WINDOW_NORMAL)
cv2.imshow('Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

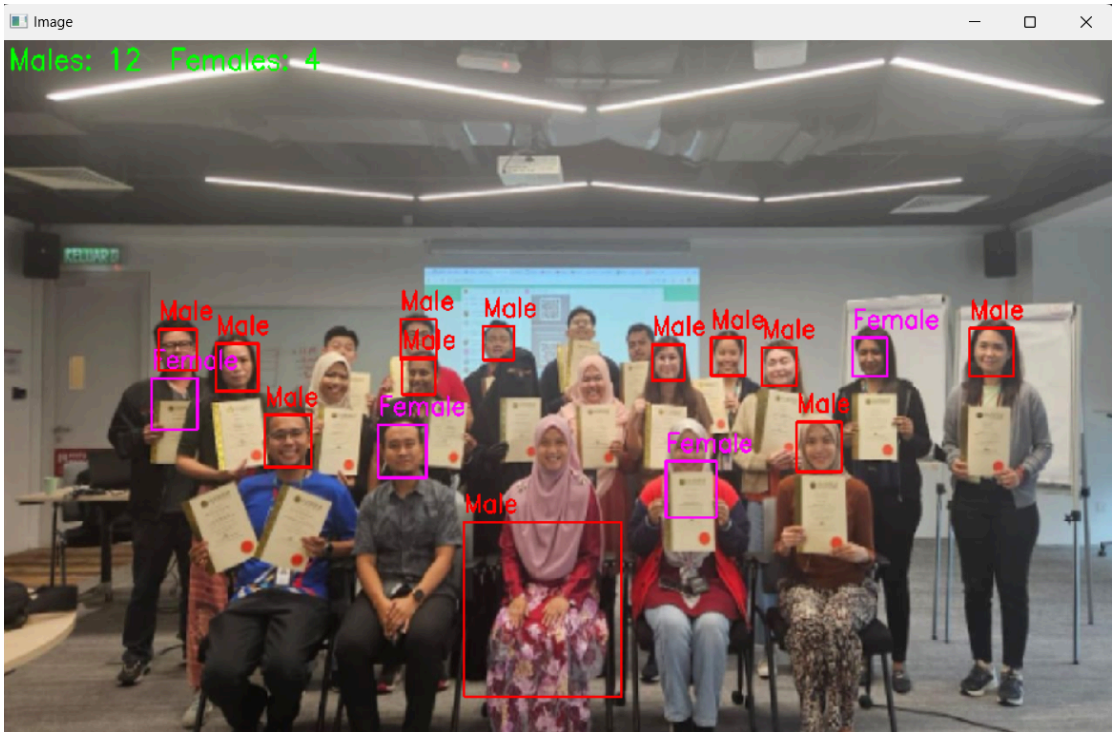
Output image from the code inference:

1) trained model ‘good_model.pt’

Result: Error and could not load the output images.

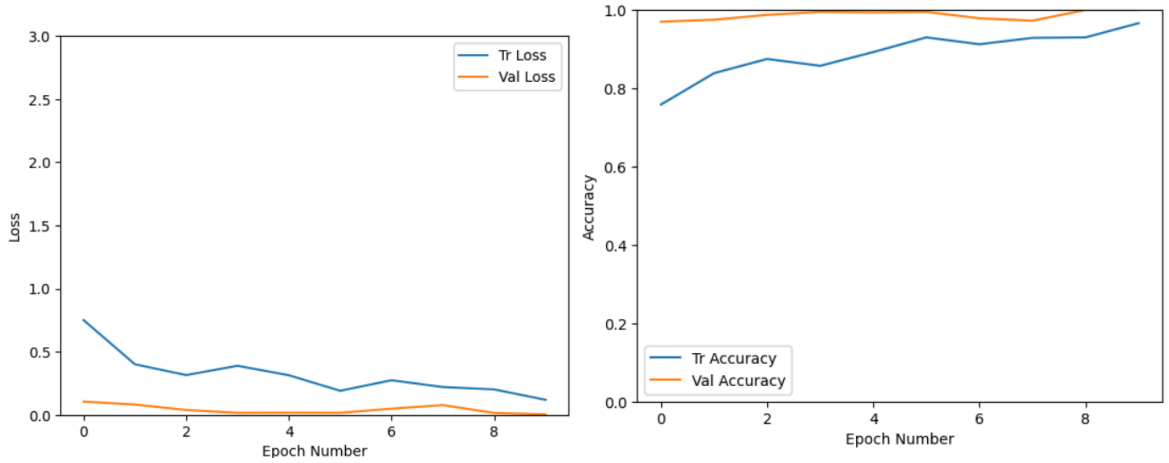
2) trained model ‘good_model (2).pt’

Result: The output image resulting in can detect and count some of both men and women but not so accurately.



```
[ ]  100%|██████████| 10/10 [04:19<00:00, 25.70s/it]
Epoch: 1/10
Epoch : 000, Training: Loss: 0.7502, Accuracy: 75.8448%,
                Validation : Loss : 0.1049, Accuracy: 96.9962%, Time: 27.1757s
Epoch: 2/10
Epoch : 001, Training: Loss: 0.4003, Accuracy: 83.8548%,
                Validation : Loss : 0.0816, Accuracy: 97.4969%, Time: 26.0692s
Epoch: 3/10
Epoch : 002, Training: Loss: 0.3148, Accuracy: 87.4844%,
                Validation : Loss : 0.0385, Accuracy: 98.7484%, Time: 25.5598s
Epoch: 4/10
Epoch : 003, Training: Loss: 0.3885, Accuracy: 85.7322%,
                Validation : Loss : 0.0163, Accuracy: 99.4994%, Time: 25.7198s
Epoch: 5/10
Epoch : 004, Training: Loss: 0.3135, Accuracy: 89.2365%,
                Validation : Loss : 0.0174, Accuracy: 99.3742%, Time: 25.5808s
Epoch: 6/10
Epoch : 005, Training: Loss: 0.1905, Accuracy: 92.9912%,
                Validation : Loss : 0.0164, Accuracy: 99.4994%, Time: 25.4141s
Epoch: 7/10
Epoch : 006, Training: Loss: 0.2740, Accuracy: 91.2390%,
                Validation : Loss : 0.0488, Accuracy: 97.8723%, Time: 26.5807s
Epoch: 8/10
Epoch : 007, Training: Loss: 0.2201, Accuracy: 92.8661%,
                Validation : Loss : 0.0775, Accuracy: 97.2466%, Time: 25.4706s
Epoch: 9/10
Epoch : 008, Training: Loss: 0.2014, Accuracy: 92.9912%,
                Validation : Loss : 0.0153, Accuracy: 100.0000%, Time: 25.5681s
Epoch: 10/10
Epoch : 009, Training: Loss: 0.1193, Accuracy: 96.6208%,
                Validation : Loss : 0.0040, Accuracy: 100.0000%, Time: 25.5038s
```

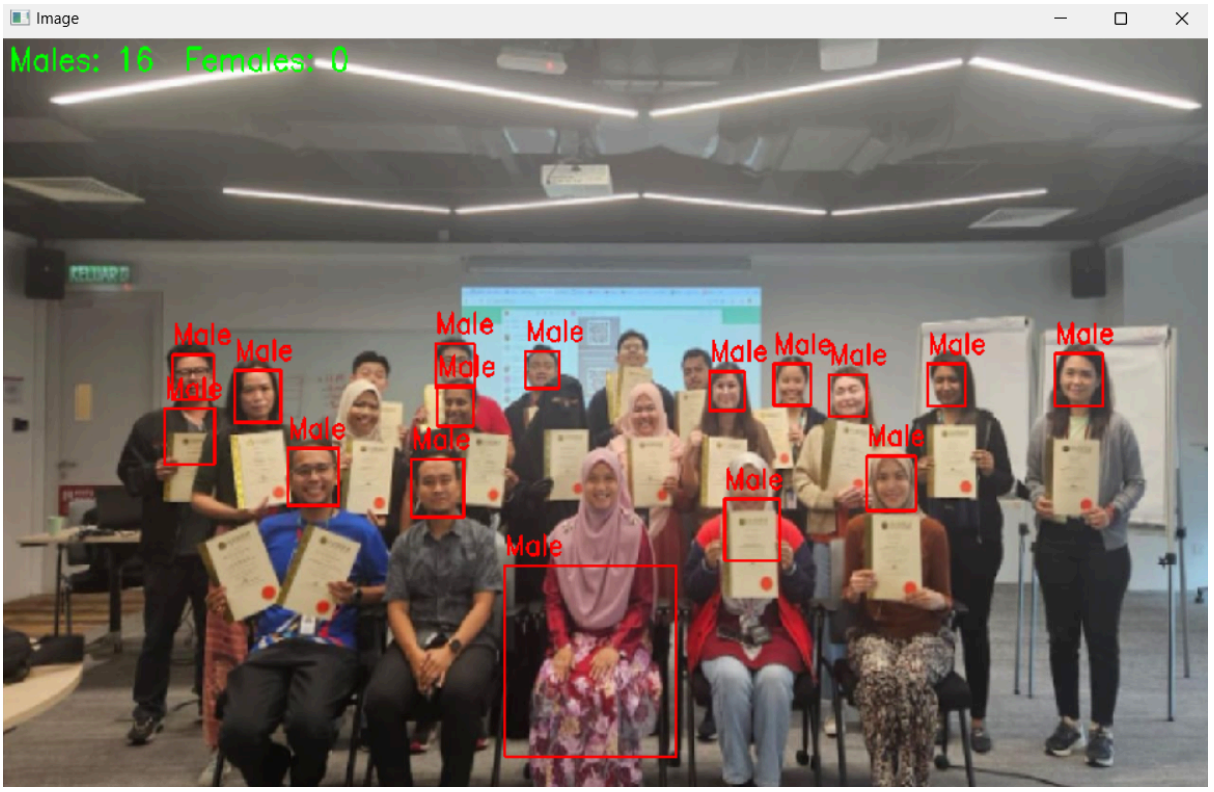
Figure 4: Train the model for 10 epochs and get good_model (2).pt



Graph good_model (2).pt

3) trained model ‘good_model(3).pt’

Result: The output image resulting in could not detect and count the women.



- GitHub link containing the code, model and dataset used for training.
  https://github.com/sfeaqida/Quiz2-Object-Detection