SOCIAL ROBOT

AN END-SEMESTER PROJECT REPORT ON THE SUBJECT OF INTRODUCTION TO AI ROBOTICS

Submitted to

Amrita Vishwa Vidyapeetham

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE ENGINEERING (AIE)

By

GURUPRASATH M R

CH.EN.U4AIE22015

Submitted to

DR. GOLAK BIHARI MAHANTA

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI – 601103

April 2024

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956

SCHOOL OF
COMPUTING
CHENNAI

BONAFIDE CERTIFICATE

Certified that this project report **SOCIAL ROBOT”** is the bonafide work of **“Guruprasath M R”** who carried out the project work under my supervision towards his completion of the end semester project for the subject “INTRODUCTION TO AI ROBOTICS (22AIE214)”.

SIGNATURE

Dr. Golak Bihari Mahanta

Course Instructor

Assistant Professor (Sr. Gr)
Dept. of Mechanical Engineering Amrita School of
Engineering, Amrita Vishwa Vidyapeetham,
Chennai

Project Overview:

The robotics project is a comprehensive system designed to interact with humans through various modalities including face recognition, speech-to-text conversion, emotion detection, and natural language processing. The project is divided into three main parts, each handled by a separate team, given below

Team-A(My self): Face Recognition

Objective: Implement a face recognition system capable of identifying known individual

Methodology: Utilize computer vision techniques and machine learning algorithms to train a model for face recognition.

Output: Upon detecting a face, the system will identify the individual if they are registered in the database.

Team-B: Speech-to-Text and Emotion Detection

Objective: Develop a system to capture speech, convert it to text, and detect emotions from the speech input.

Methodology: Employ speech recognition algorithms to transcribe spoken words into text. Additionally, utilize sentiment analysis techniques to detect emotions such as happiness, sadness, excitement, etc.

Output: The system will provide both the transcribed text and the detected emotions from the speech input.

Team-C: UI

Objective: Design a user interface to display the emotions detected by Team-B's system.

Methodology: Create an interactive web interface using HTML, CSS, and JavaScript, Integrated with Flask for backend functionality. The interface will dynamically showcase detected emotions through text and emojis.

Output: The interface will provide users with a visual representation of emotions, along with options for manual input of emotions and speech recording.

MY PART(TEAM A – FACE RECOGNITION)

My responsibility involves creating the preliminary code for extracting the video feed from the frontend, extracting text, emotions, and queries, and feeding them into the NLP part. Shyam has developed the NLP component and received the output, which is then passed to the frontend. Next, Pradeep optimized my code to seamlessly integrate it into the frontend. all codes and my trails are uploaded in this GitHub Repo "https://github.com/gru13/Robotics-project"

Dataset Creation

To create a dataset for face recognition using images captured from a webcam, you can utilize the OpenCV library in Python. OpenCV provides a comprehensive set of tools for image processing and computer vision tasks. First, set up your Python environment with OpenCV installed. Then, write a script to capture images from the webcam and save them along with labels corresponding to the identities of the individuals in the images.

Steps involved

- Import Libraries: Import OpenCV and OS libraries.

- Initialize Webcam: Use OpenCV to initialize the webcam and start capturing frames.

- Capture Images: Capture successive image frames from the webcam with a 5-millisecond interva between each frame.

- Save Images: Once a face is detected, save the cropped face region along with its corresponding labe (e.g., person's name or ID) into a dataset directory.

- Dataset Organization: Organize the dataset directory structure with subdirectories for each individual containing their respective images.

Face Detection

Initially, I and Deepak attempted to utilize OpenCV's pre-trained Haar cascades with increments in image quantity: starting from 30 images, then 50, 100, 200, and finally 500. However, we encountered issues during testing, achieving only a 50% accuracy rate. Upon receiving a suggestion from my friend Neelraj, we decided to discontinue its use and instead transitioned to employing YOLO v8, achieving a 98% accuracy rate.

Emotion Detection

For emotion detection, I utilized the deepface module in Python, which includes built-in modules for pre-trained emotional detection models.

Voice Recognition

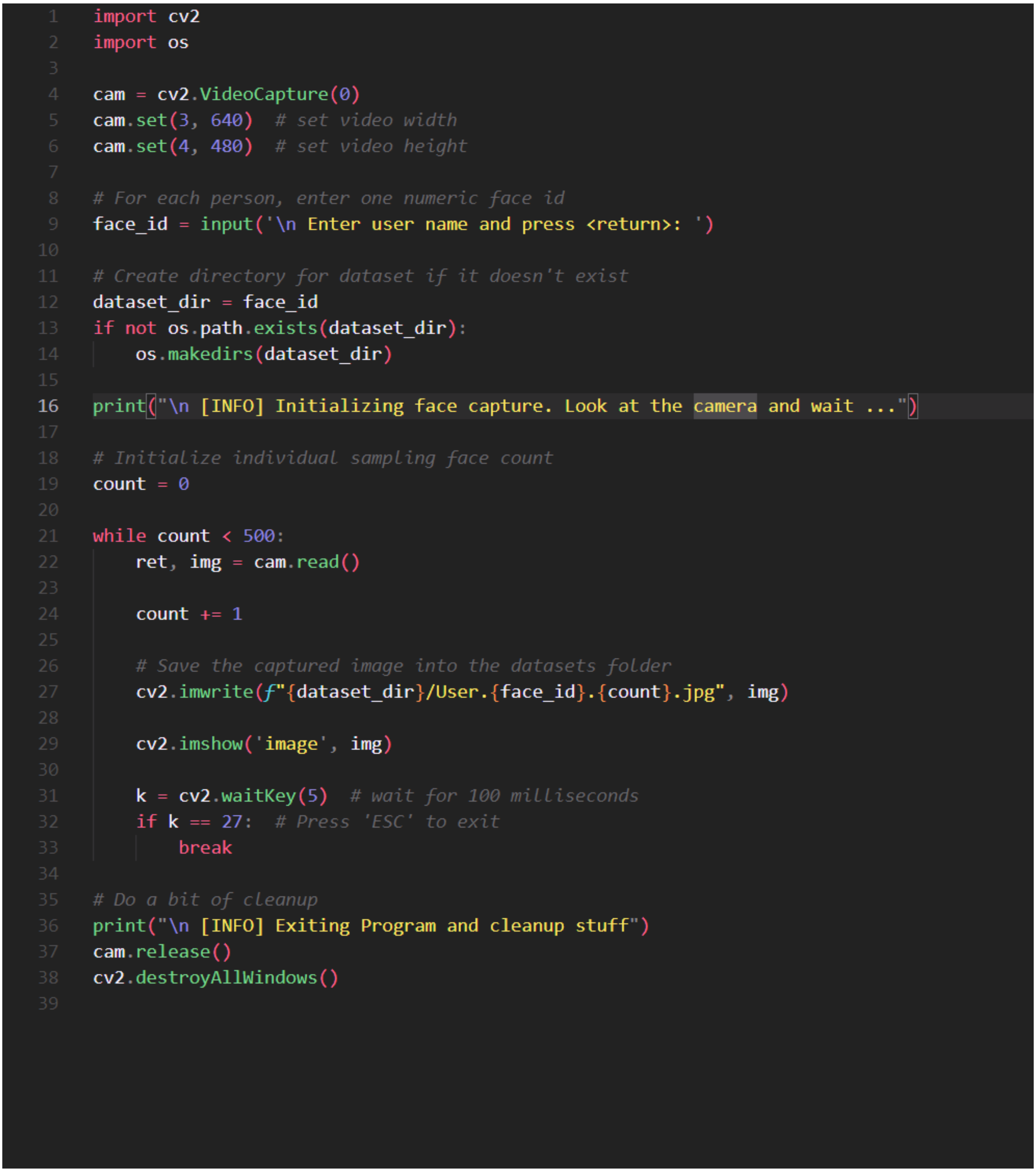
Here, we utilized OpenAI's Whisper model for voice recognition.

Integration Process

Here I combine the above 3 process in a function and NLP function from shyam and return to frontend

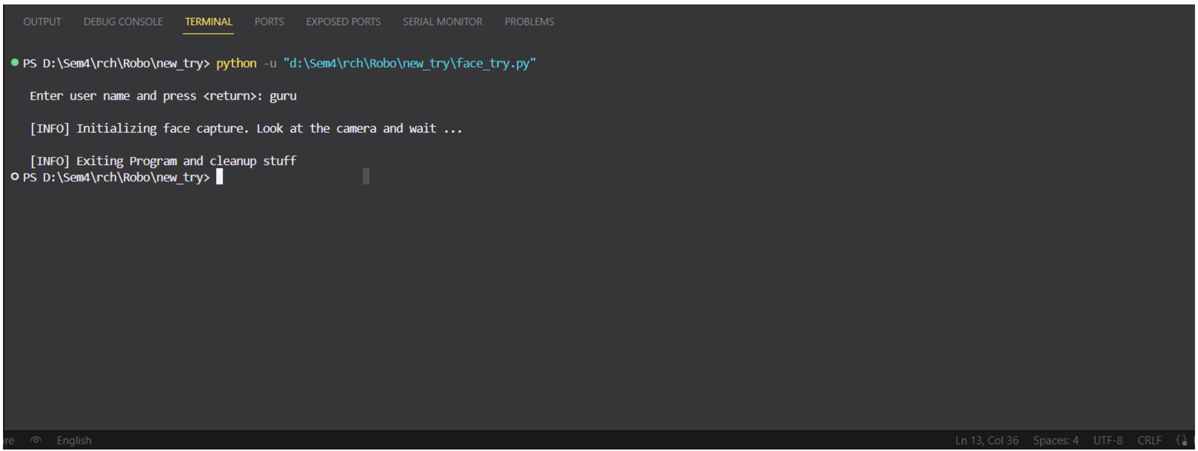
Code

Dataset creation

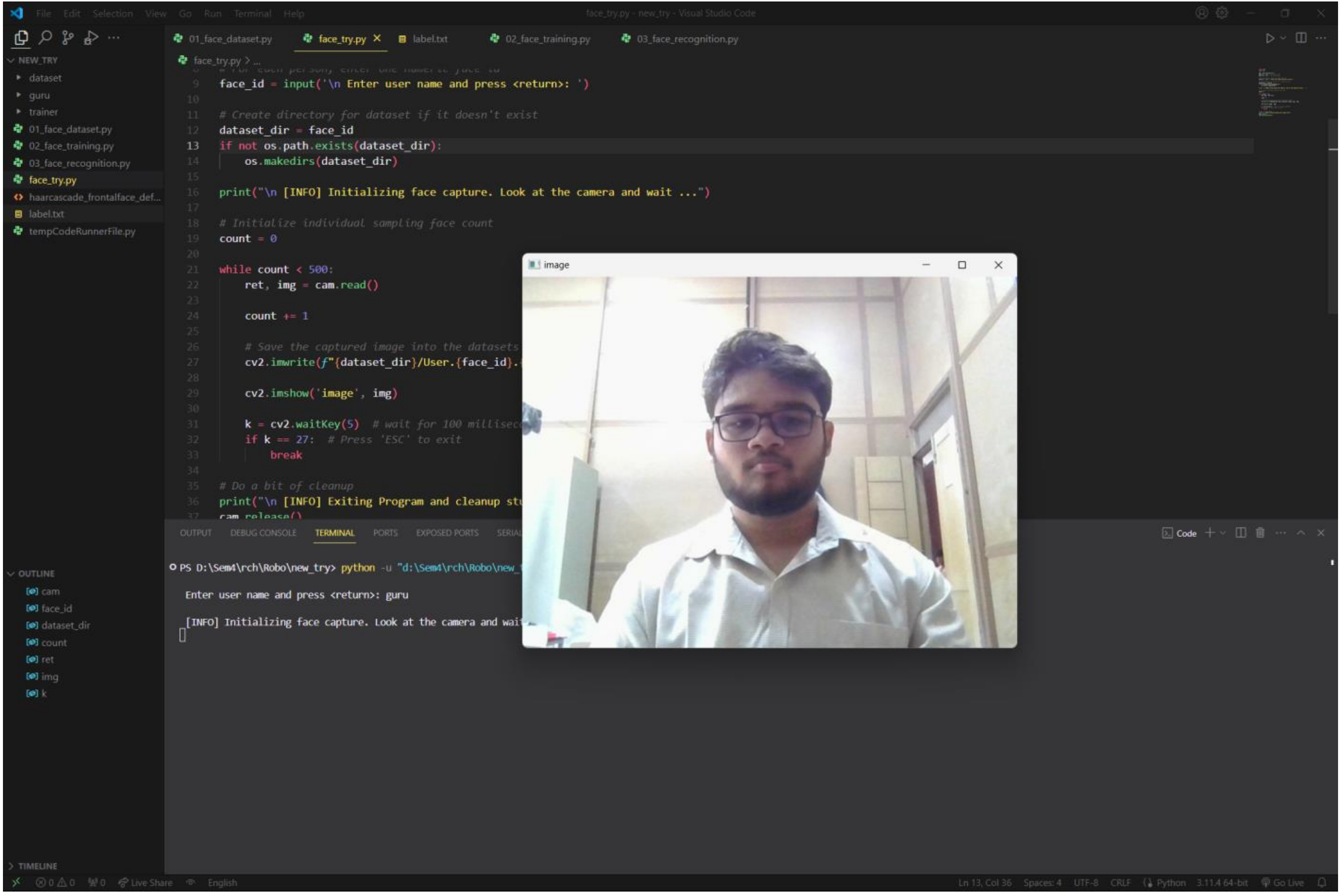


```python
import cv2
import os

cam = cv2.VideoCapture(0)
cam.set(3, 640)  # set video width
cam.set(4, 480)  # set video height

# For each person, enter one numeric face id
face_id = input('\n Enter user name and press <return>: ')

# Create directory for dataset if it doesn't exist
dataset_dir = face_id
if not os.path.exists(dataset_dir):
    os.makedirs(dataset_dir)

print("\n [INFO] Initializing face capture. Look at the camera and wait ...")

# Initialize individual sampling face count
count = 0

while count < 500:
    ret, img = cam.read()

    count += 1

    # Save the captured image into the datasets folder
    cv2.imwrite(f"{dataset_dir}/User.{face_id}.{count}.jpg", img)

    cv2.imshow('image', img)

    k = cv2.waitKey(5)  # wait for 100 milliseconds
    if k == 27:  # Press 'ESC' to exit
        break

# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

Input and Output

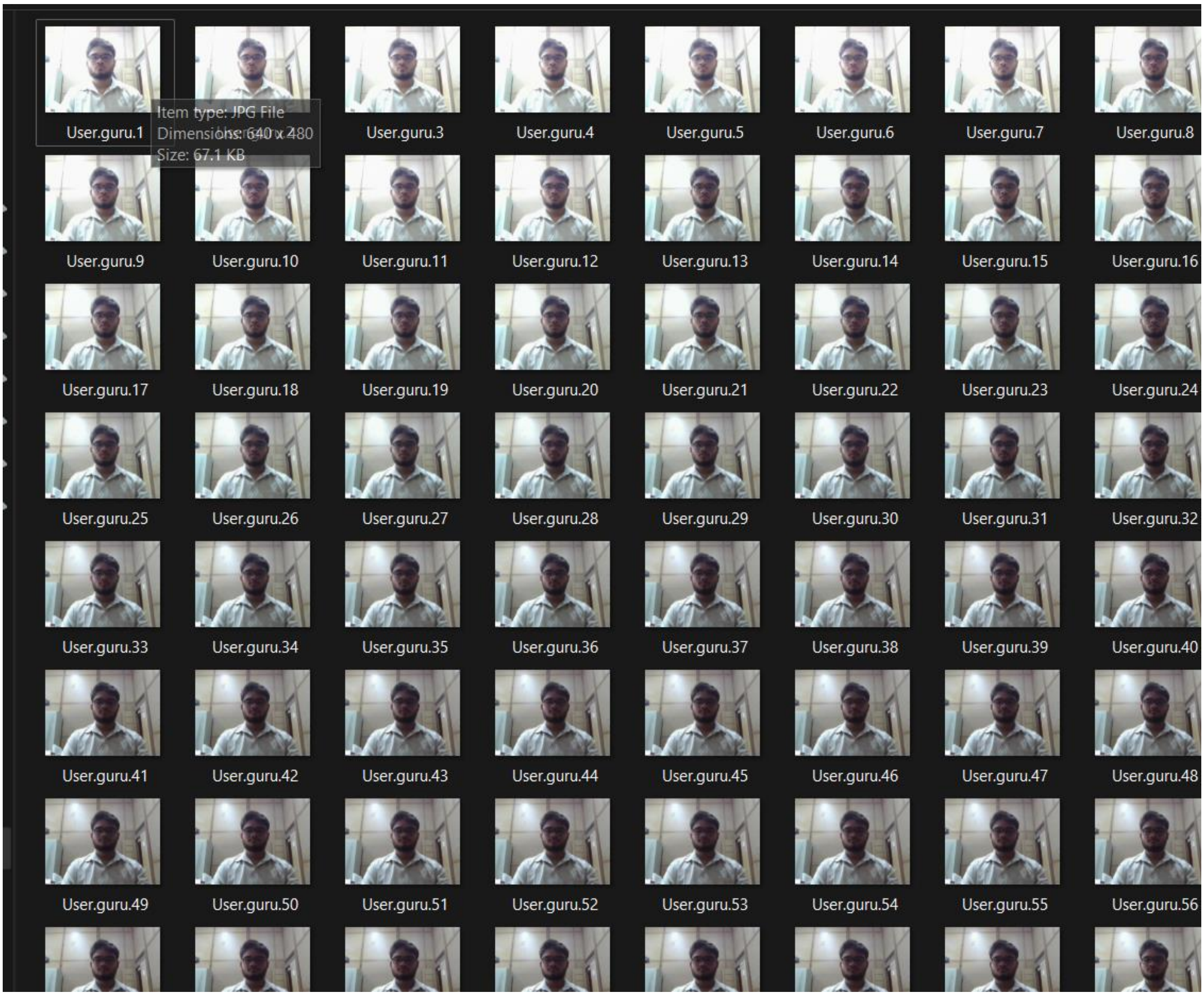


```
● PS D:\Sem4\rch\Robo\new_try> python -u "d:\Sem4\rch\Robo\new_try\face_try.py"

  Enter user name and press <return>: guru

  [INFO] Initializing face capture. Look at the camera and wait ...

  [INFO] Exiting Program and cleanup stuff
○ PS D:\Sem4\rch\Robo\new_try> 
```

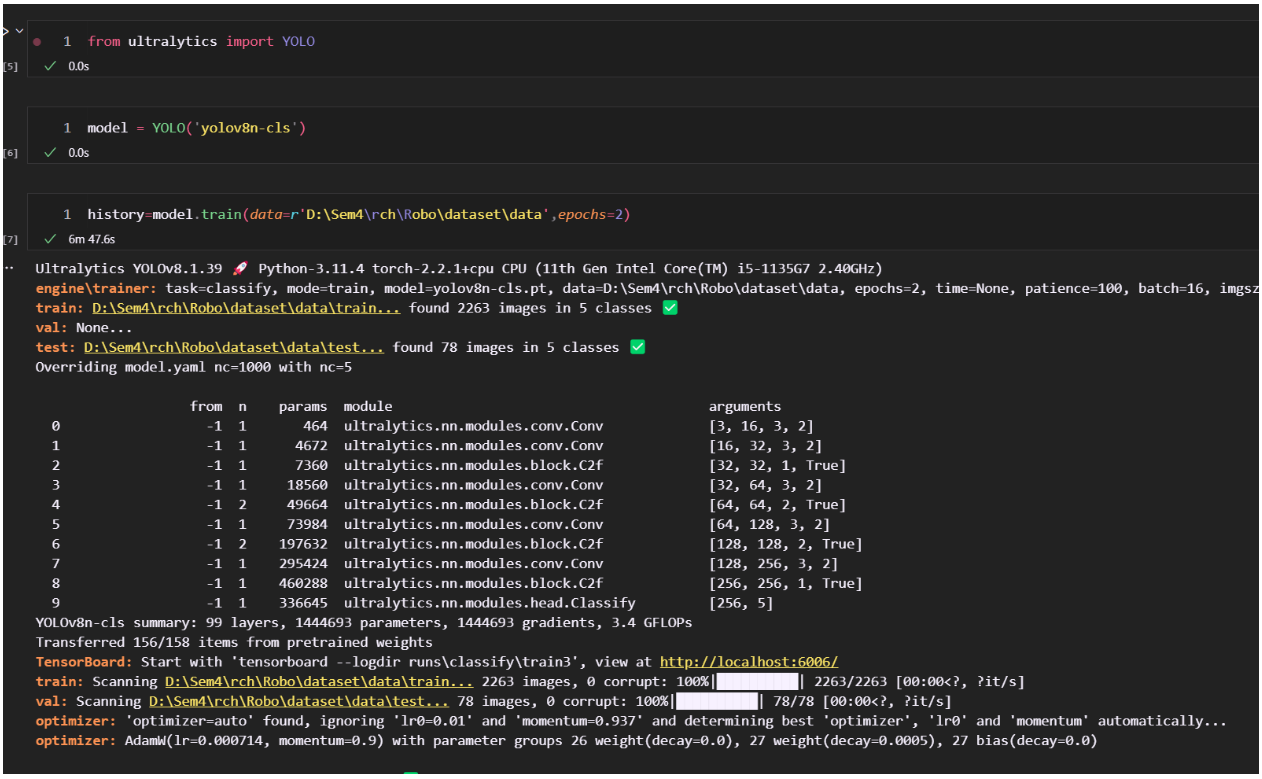


```python
face_id = input('\n Enter user name and press <return>: ')

# Create directory for dataset if it doesn't exist
dataset_dir = face_id
if not os.path.exists(dataset_dir):
    os.makedirs(dataset_dir)

print("\n [INFO] Initializing face capture. Look at the camera and wait ...")

# Initialize individual sampling face count
count = 0

while count < 500:
    ret, img = cam.read()

    count += 1

    # Save the captured image into the datasets
    cv2.imwrite(f"{dataset_dir}/User.{face_id}.

    cv2.imshow('image', img)

    k = cv2.waitKey(5)  # wait for 100 millisec
    if k == 27:  # Press 'ESC' to exit
        break

# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stu
cam.release()
```

```
○ PS D:\Sem4\rch\Robo\new_try> python -u "d:\Sem4\rch\Robo\new_

  Enter user name and press <return>: guru

  [INFO] Initializing face capture. Look at the camera and wai
```



Face Detection

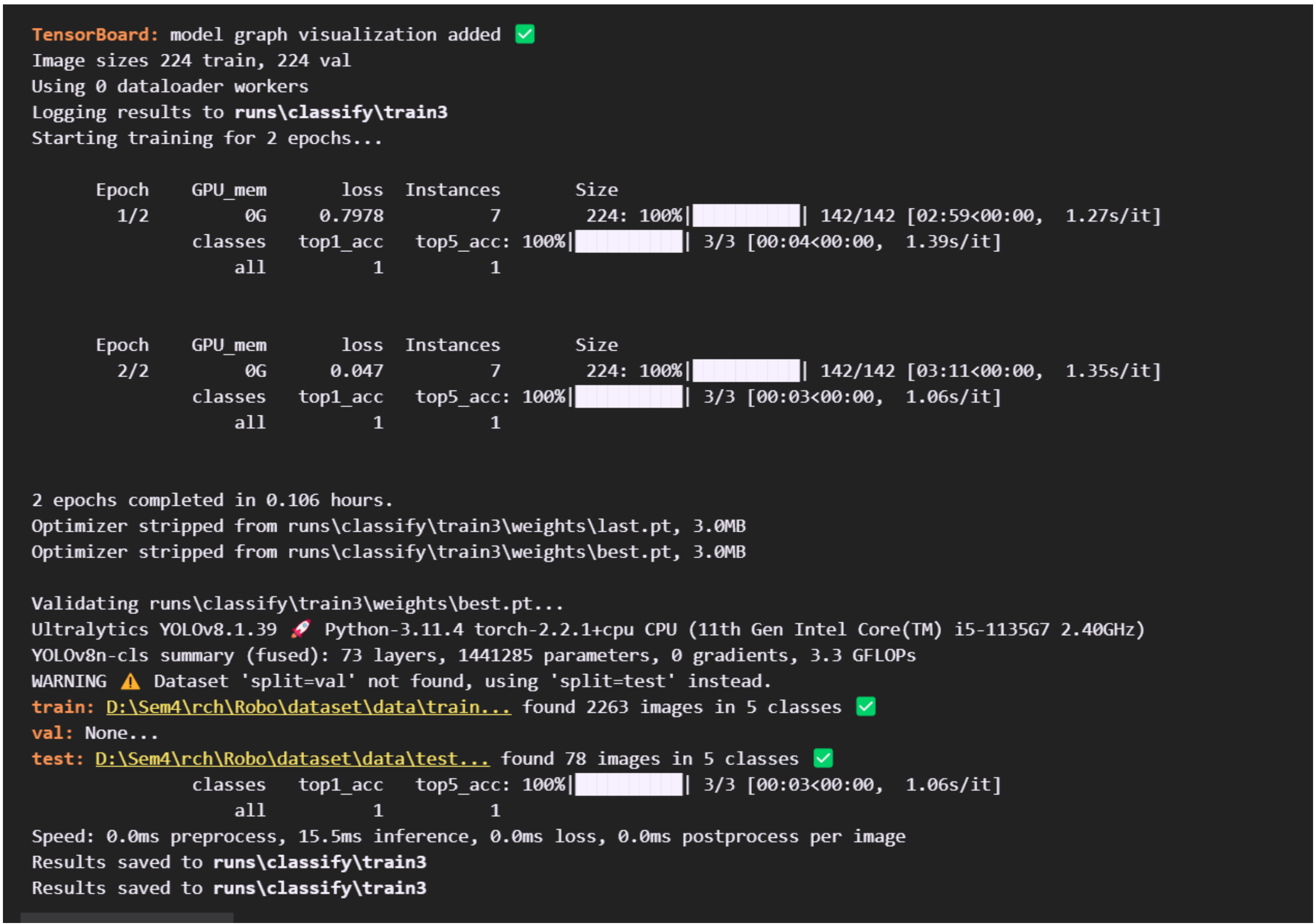


```
from ultralytics import YOLO
```

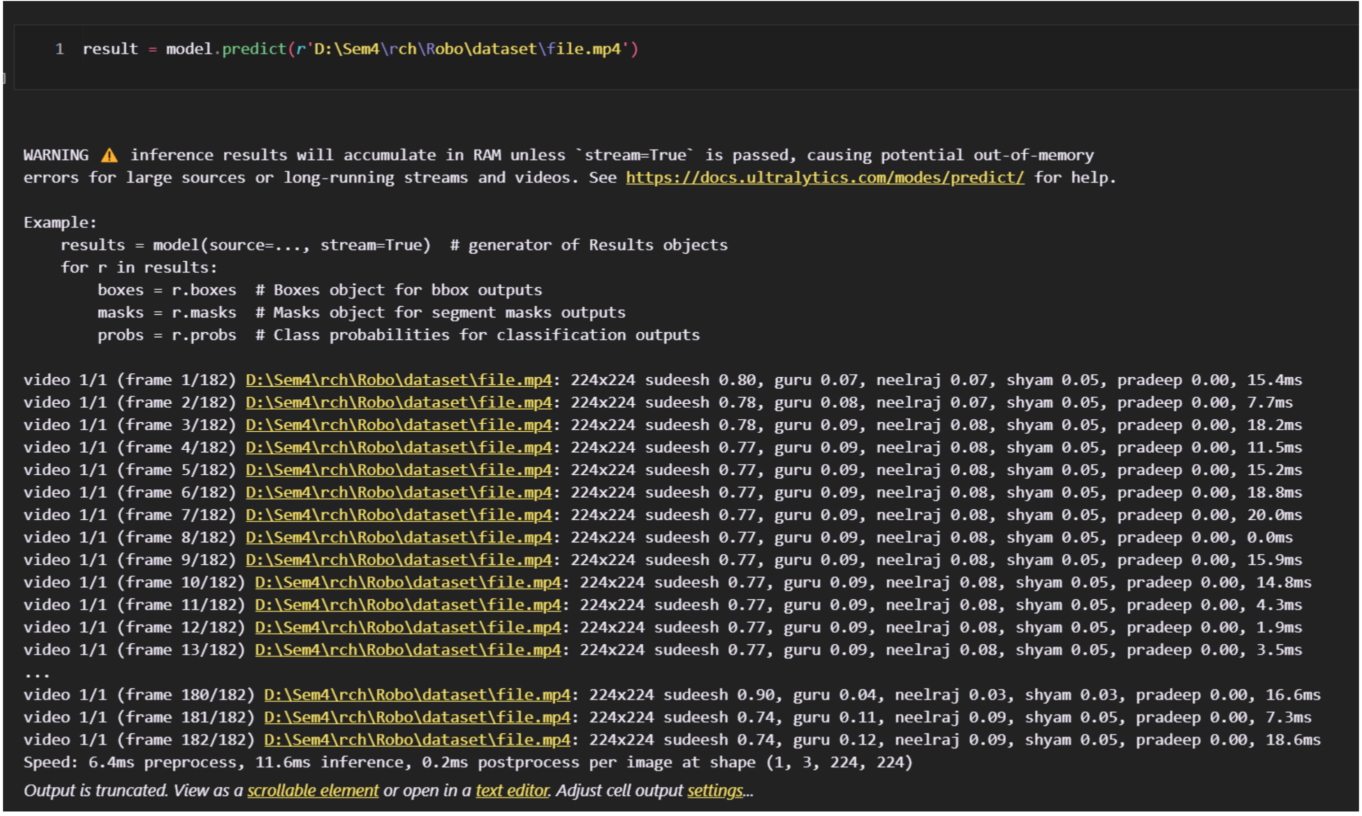
```
model = YOLO('yolov8n-cls')
```

```
history=model.train(data=r'D:\Sem4\rch\Robo\dataset\data',epochs=2)
```

```
Ultralytics YOLOv8.1.39 🚀 Python-3.11.4 torch-2.2.1+cpu CPU (11th Gen Intel Core(TM) i5-1135G7 2.40GHz)
engine\trainer: task=classify, mode=train, model=yolov8n-cls.pt, data=D:\Sem4\rch\Robo\dataset\data, epochs=2, time=None, patience=100, batch=16, imgsz
train: D:\Sem4\rch\Robo\dataset\data\train... found 2263 images in 5 classes ✅
val: None...
test: D:\Sem4\rch\Robo\dataset\data\test... found 78 images in 5 classes ✅
Overriding model.yaml nc=1000 with nc=5

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    336645  ultralytics.nn.modules.head.Classify         [256, 5]
YOLOv8n-cls summary: 99 layers, 1444693 parameters, 1444693 gradients, 3.4 GFLOPs
Transferred 156/158 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs\classify\train3', view at http://localhost:6006/
train: Scanning D:\Sem4\rch\Robo\dataset\data\train... 2263 images, 0 corrupt: 100%|██████████| 2263/2263 [00:00<?, ?it/s]
val: Scanning D:\Sem4\rch\Robo\dataset\data\test... 78 images, 0 corrupt: 100%|██████████| 78/78 [00:00<?, ?it/s]
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.000714, momentum=0.9) with parameter groups 26 weight(decay=0.0), 27 weight(decay=0.0005), 27 bias(decay=0.0)
```



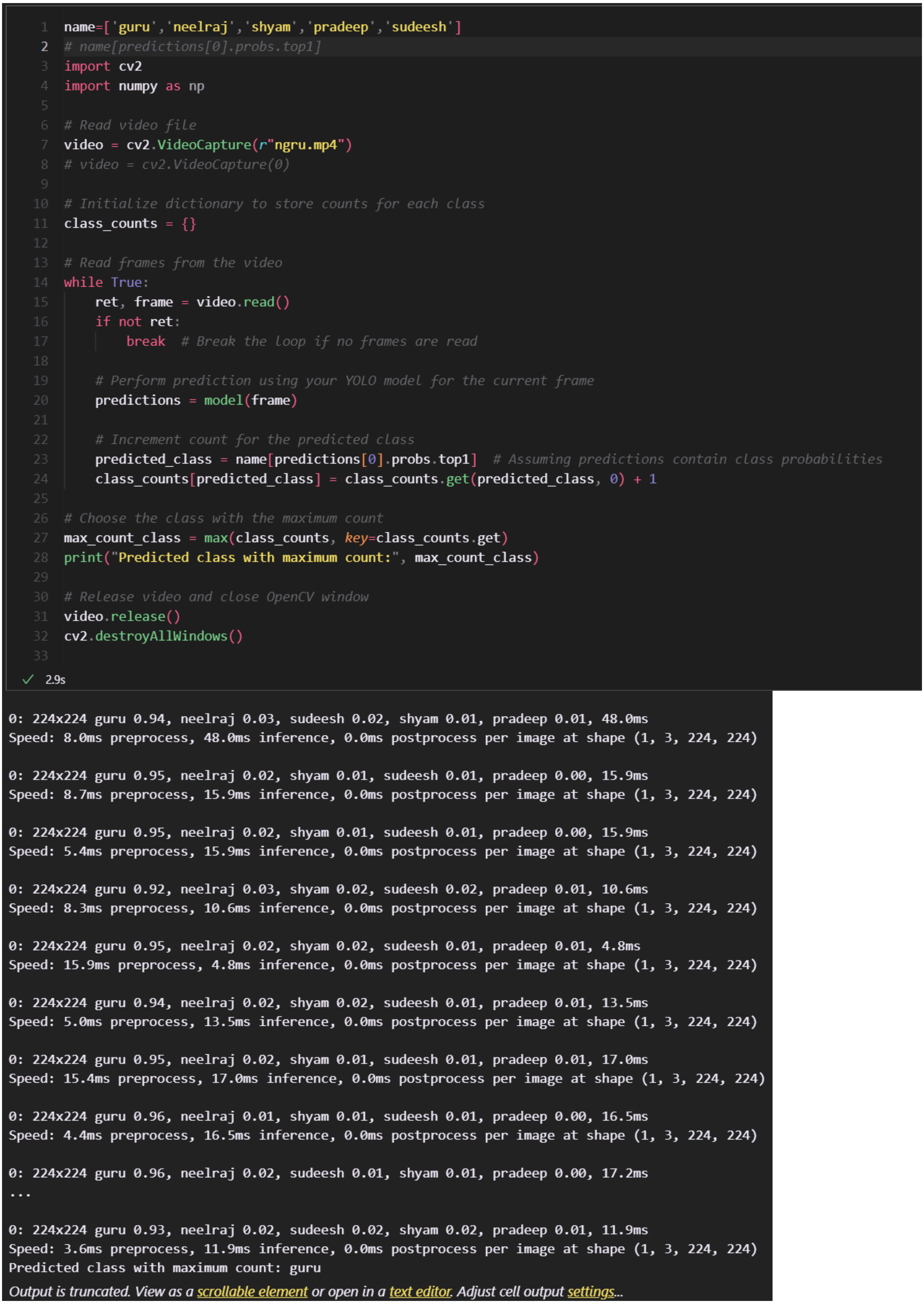
```
TensorBoard: model graph visualization added ✅
Image sizes 224 train, 224 val
Using 0 dataloader workers
Logging results to runs\classify\train3
Starting training for 2 epochs...

      Epoch    GPU_mem       loss  Instances       Size
        1/2         0G     0.7978          7        224: 100%|██████████| 142/142 [02:59<00:00,  1.27s/it]
                classes   top1_acc   top5_acc: 100%|██████████| 3/3 [00:04<00:00,  1.39s/it]
                    all          1          1


      Epoch    GPU_mem       loss  Instances       Size
        2/2         0G      0.047          7        224: 100%|██████████| 142/142 [03:11<00:00,  1.35s/it]
                classes   top1_acc   top5_acc: 100%|██████████| 3/3 [00:03<00:00,  1.06s/it]
                    all          1          1


2 epochs completed in 0.106 hours.
Optimizer stripped from runs\classify\train3\weights\last.pt, 3.0MB
Optimizer stripped from runs\classify\train3\weights\best.pt, 3.0MB

Validating runs\classify\train3\weights\best.pt...
Ultralytics YOLOv8.1.39 🚀 Python-3.11.4 torch-2.2.1+cpu CPU (11th Gen Intel Core(TM) i5-1135G7 2.40GHz)
YOLOv8n-cls summary (fused): 73 layers, 1441285 parameters, 0 gradients, 3.3 GFLOPs
WARNING ⚠️ Dataset 'split=val' not found, using 'split=test' instead.
train: D:\Sem4\rch\Robo\dataset\data\train... found 2263 images in 5 classes ✅
val: None...
test: D:\Sem4\rch\Robo\dataset\data\test... found 78 images in 5 classes ✅
                classes   top1_acc   top5_acc: 100%|██████████| 3/3 [00:03<00:00,  1.06s/it]
                    all          1          1
Speed: 0.0ms preprocess, 15.5ms inference, 0.0ms loss, 0.0ms postprocess per image
Results saved to runs\classify\train3
Results saved to runs\classify\train3
```



```python
result = model.predict(r'D:\Sem4\rch\Robo\dataset\file.mp4')
```

```
WARNING ⚠️ inference results will accumulate in RAM unless `stream=True` is passed, causing potential out-of-memory
errors for large sources or long-running streams and videos. See https://docs.ultralytics.com/modes/predict/ for help.

Example:
    results = model(source=..., stream=True)  # generator of Results objects
    for r in results:
        boxes = r.boxes  # Boxes object for bbox outputs
        masks = r.masks  # Masks object for segment masks outputs
        probs = r.probs  # Class probabilities for classification outputs

video 1/1 (frame 1/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.80, guru 0.07, neelraj 0.07, shyam 0.05, pradeep 0.00, 15.4ms
video 1/1 (frame 2/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.78, guru 0.08, neelraj 0.07, shyam 0.05, pradeep 0.00, 7.7ms
video 1/1 (frame 3/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.78, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 18.2ms
video 1/1 (frame 4/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 11.5ms
video 1/1 (frame 5/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 15.2ms
video 1/1 (frame 6/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 18.8ms
video 1/1 (frame 7/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 20.0ms
video 1/1 (frame 8/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 0.0ms
video 1/1 (frame 9/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 15.9ms
video 1/1 (frame 10/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 14.8ms
video 1/1 (frame 11/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 4.3ms
video 1/1 (frame 12/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 1.9ms
video 1/1 (frame 13/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.77, guru 0.09, neelraj 0.08, shyam 0.05, pradeep 0.00, 3.5ms
...
video 1/1 (frame 180/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.90, guru 0.04, neelraj 0.03, shyam 0.03, pradeep 0.00, 16.6ms
video 1/1 (frame 181/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.74, guru 0.11, neelraj 0.09, shyam 0.05, pradeep 0.00, 7.3ms
video 1/1 (frame 182/182) D:\Sem4\rch\Robo\dataset\file.mp4: 224x224 sudeesh 0.74, guru 0.12, neelraj 0.09, shyam 0.05, pradeep 0.00, 18.6ms
Speed: 6.4ms preprocess, 11.6ms inference, 0.2ms postprocess per image at shape (1, 3, 224, 224)
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



```
name=['guru','neelraj','shyam','pradeep','sudeesh']
# name[predictions[0].probs.top1]
import cv2
import numpy as np

# Read video file
video = cv2.VideoCapture(r"ngru.mp4")
# video = cv2.VideoCapture(0)

# Initialize dictionary to store counts for each class
class_counts = {}

# Read frames from the video
while True:
    ret, frame = video.read()
    if not ret:
        break  # Break the loop if no frames are read

    # Perform prediction using your YOLO model for the current frame
    predictions = model(frame)

    # Increment count for the predicted class
    predicted_class = name[predictions[0].probs.top1]  # Assuming predictions contain class probabilities
    class_counts[predicted_class] = class_counts.get(predicted_class, 0) + 1

# Choose the class with the maximum count
max_count_class = max(class_counts, key=class_counts.get)
print("Predicted class with maximum count:", max_count_class)

# Release video and close OpenCV window
video.release()
cv2.destroyAllWindows()

```

```
0: 224x224 guru 0.94, neelraj 0.03, sudeesh 0.02, shyam 0.01, pradeep 0.01, 48.0ms
Speed: 8.0ms preprocess, 48.0ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.95, neelraj 0.02, shyam 0.01, sudeesh 0.01, pradeep 0.00, 15.9ms
Speed: 8.7ms preprocess, 15.9ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.95, neelraj 0.02, shyam 0.01, sudeesh 0.01, pradeep 0.00, 15.9ms
Speed: 5.4ms preprocess, 15.9ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.92, neelraj 0.03, shyam 0.02, sudeesh 0.02, pradeep 0.01, 10.6ms
Speed: 8.3ms preprocess, 10.6ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.95, neelraj 0.02, shyam 0.02, sudeesh 0.01, pradeep 0.01, 4.8ms
Speed: 15.9ms preprocess, 4.8ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.94, neelraj 0.02, shyam 0.02, sudeesh 0.01, pradeep 0.01, 13.5ms
Speed: 5.0ms preprocess, 13.5ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.95, neelraj 0.02, shyam 0.01, sudeesh 0.01, pradeep 0.01, 17.0ms
Speed: 15.4ms preprocess, 17.0ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.96, neelraj 0.01, shyam 0.01, sudeesh 0.01, pradeep 0.00, 16.5ms
Speed: 4.4ms preprocess, 16.5ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)

0: 224x224 guru 0.96, neelraj 0.02, sudeesh 0.01, shyam 0.01, pradeep 0.00, 17.2ms
...

0: 224x224 guru 0.93, neelraj 0.02, sudeesh 0.02, shyam 0.02, pradeep 0.01, 11.9ms
Speed: 3.6ms preprocess, 11.9ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)
Predicted class with maximum count: guru
```

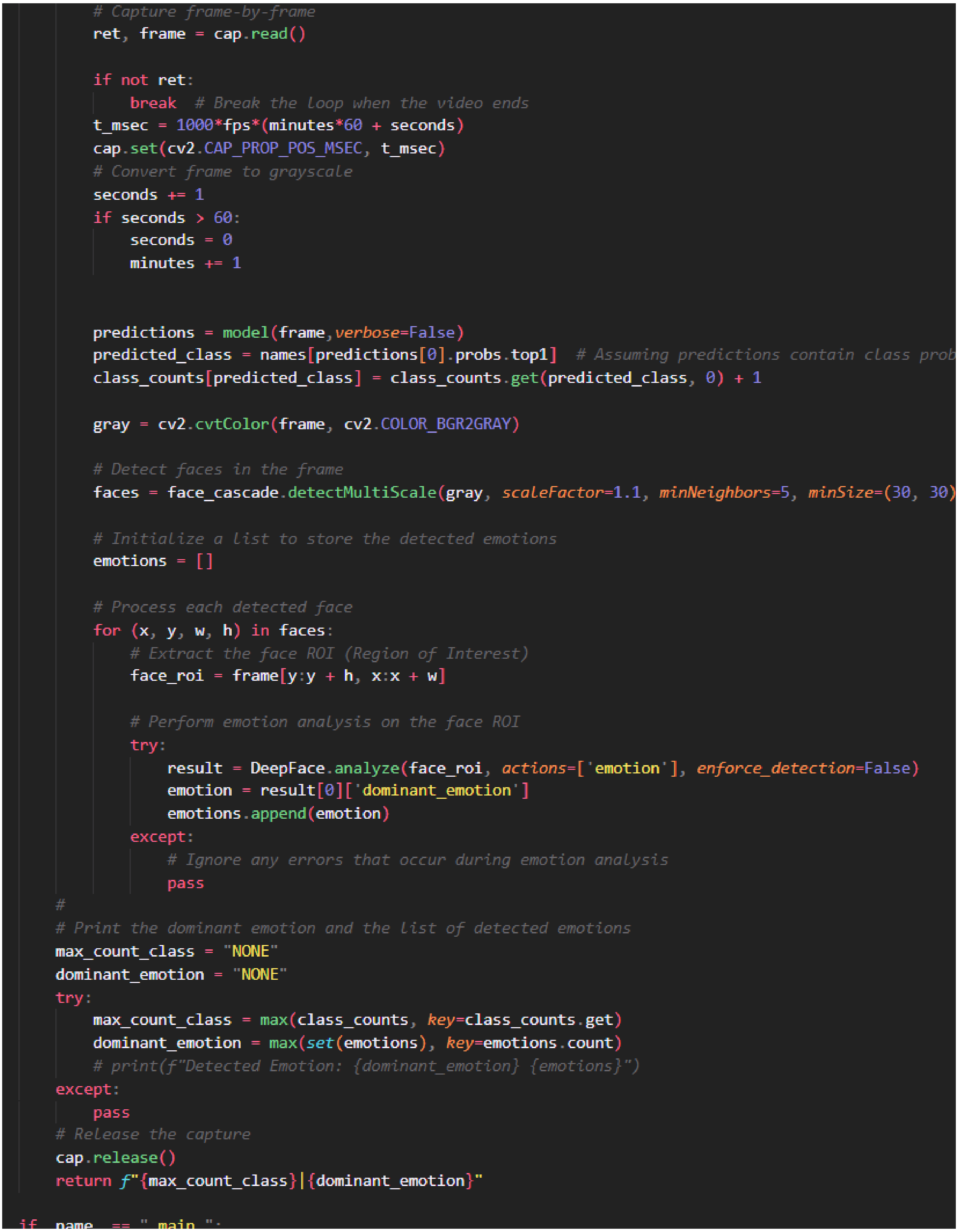
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

video Used to test

Label :Guru

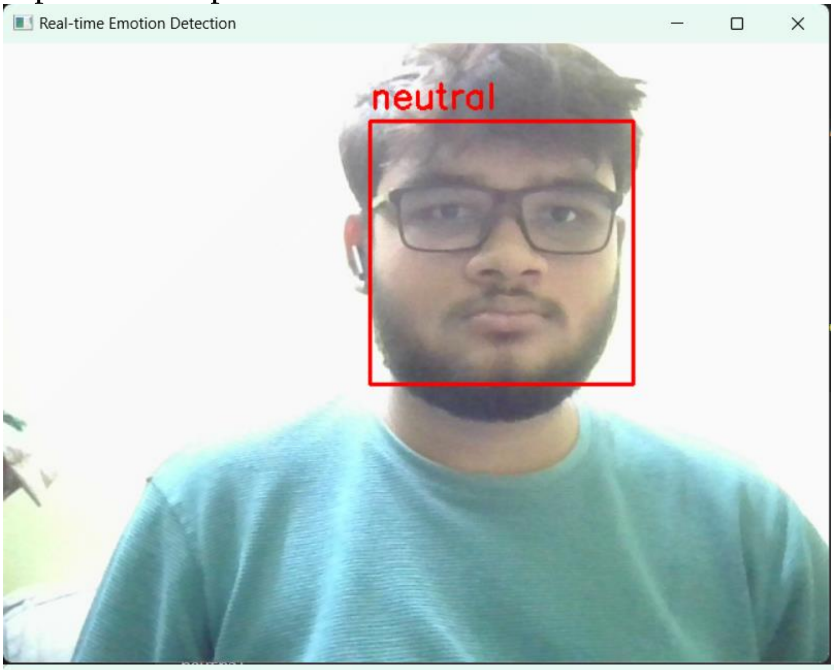
Emotion Recongintion

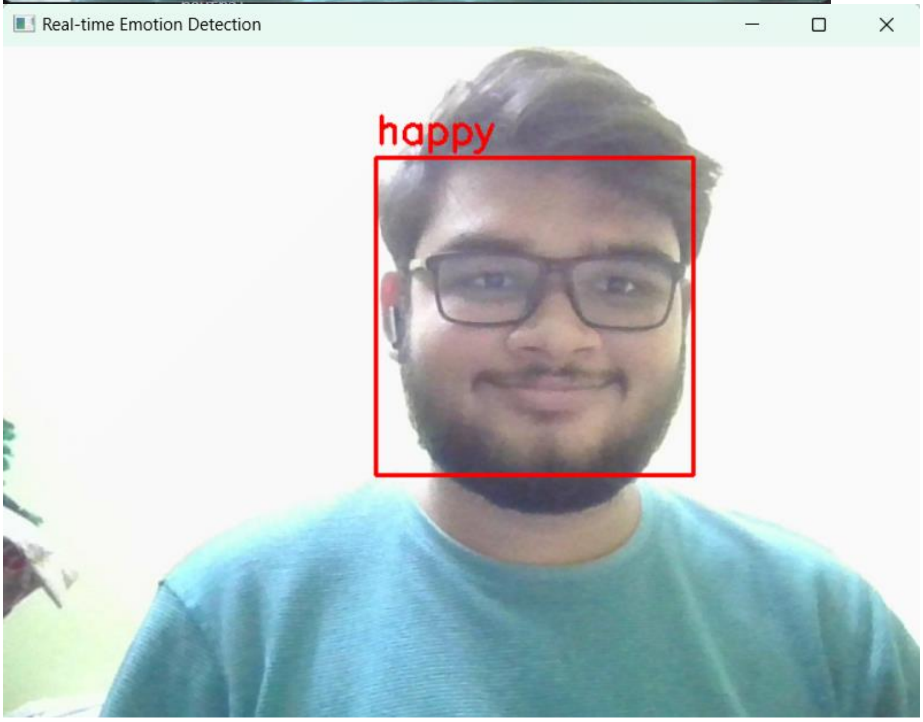
Code



```
        # Capture frame-by-frame
        ret, frame = cap.read()

        if not ret:
            break  # Break the loop when the video ends
        t_msec = 1000*fps*(minutes*60 + seconds)
        cap.set(cv2.CAP_PROP_POS_MSEC, t_msec)
        # Convert frame to grayscale
        seconds += 1
        if seconds > 60:
            seconds = 0
            minutes += 1


        predictions = model(frame,verbose=False)
        predicted_class = names[predictions[0].probs.top1]  # Assuming predictions contain class prob
        class_counts[predicted_class] = class_counts.get(predicted_class, 0) + 1

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

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

        # Initialize a list to store the detected emotions
        emotions = []

        # Process each detected face
        for (x, y, w, h) in faces:
            # Extract the face ROI (Region of Interest)
            face_roi = frame[y:y + h, x:x + w]

            # Perform emotion analysis on the face ROI
            try:
                result = DeepFace.analyze(face_roi, actions=['emotion'], enforce_detection=False)
                emotion = result[0]['dominant_emotion']
                emotions.append(emotion)
            except:
                # Ignore any errors that occur during emotion analysis
                pass
    #
    # Print the dominant emotion and the list of detected emotions
    max_count_class = "NONE"
    dominant_emotion = "NONE"
    try:
        max_count_class = max(class_counts, key=class_counts.get)
        dominant_emotion = max(set(emotions), key=emotions.count)
        # print(f"Detected Emotion: {dominant_emotion} {emotions}")
    except:
        pass
    # Release the capture
    cap.release()
    return f"{max_count_class}|{dominant_emotion}"

if __name__ == "__main__":
```

Input and Output


Real-time Emotion Detection
neutral


Real-time Emotion Detection
happy