2023

Project Report: Face Recognition Attendance System

MADE BY:

-YASH CHAUHAN 10TH RAMAN

Contents

Abstract

The Face Recognition Attendance System is an AI-powered application that utilizes computer vision and deep learning techniques to automate the process of taking attendance in various settings. The system detects and recognizes faces in real-time using a pre-trained face cascade classifier and a TensorFlow.js model created using Teachable Machine. It then records attendance information, including the person's name and timestamp, and stores it in a CSV file for future reference. This project demonstrates the practical application of AI and computer vision in streamlining routine administrative tasks.

Introduction

Taking attendance is a common task in educational institutions and workplaces. However, manual attendance tracking can be time-consuming and prone to errors. The Face Recognition Attendance System aims to address this challenge by employing AI-based facial recognition technology to accurately identify individuals and record their attendance. The system utilizes a combination of face detection, face recognition, and database management techniques to achieve its objectives.

Technologies Used

- Python

- OpenCV (cv2)

- TensorFlow (tf)

- CSV file handling

System Architecture

The Face Recognition Attendance System is built upon the following components:

1. **Face Detection**: The system uses a pre-trained Haar Cascade Classifier provided by OpenCV to detect faces within a video stream.

2. **Face Recognition**: A TensorFlow.js model exported from Teachable Machine is loaded to recognize the detected faces and assign them labels corresponding to individual names.

3. **Attendance Recording**: An attendance record is maintained as a dictionary in memory. Detected faces are recognized, and attendance information is stored in this dictionary, including names and timestamps.

4. **User Interface**: The system provides a real-time video feed with bounding boxes and labels drawn around recognized faces, making it user-friendly and visually informative.

5. **CSV File Handling**: Upon user termination (pressing 'q'), the attendance information is written to a CSV file for permanent storage.

Implementation Details

- ❖ The system captures video from the default camera (index 0) and processes each frame.
- ❖ Detected faces are resized and pre-processed to match the input shape expected by the Teachable Machine model.
- ❖ The model predicts the label (person's identity) and confidence for each detected face.
- ❖ If the confidence is above a certain threshold (0.5), the system marks the person's attendance with their name and a timestamp.
- ❖ Bounding boxes and labels are drawn on the frame to visualize the recognition process.
- ❖ The CSV file is created and attendance data is saved in it upon the user's termination command.

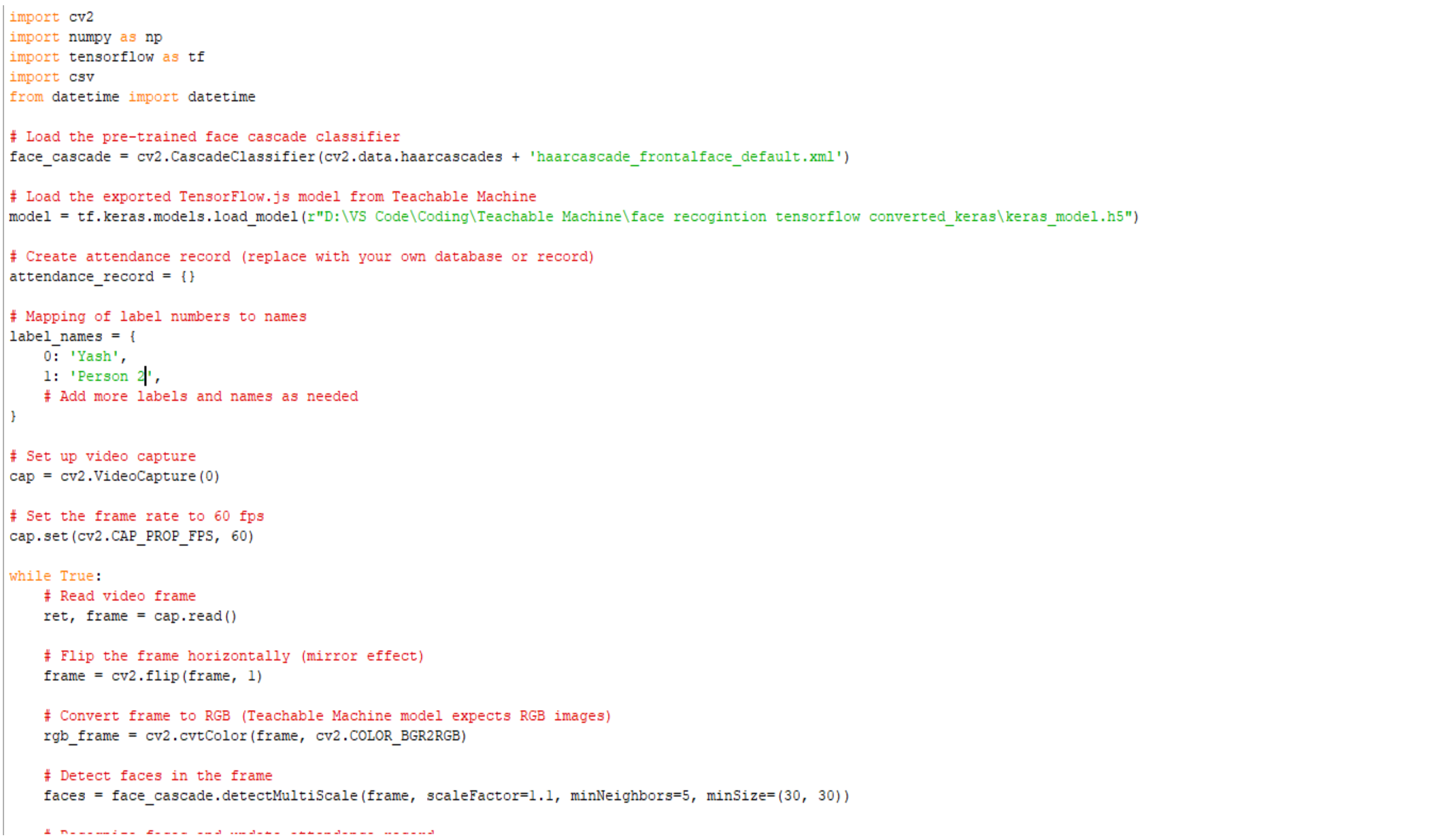
Conclusion

The Face Recognition Attendance System showcases the practical application of AI and computer vision in automating administrative tasks. By leveraging pre-trained models, the system detects and recognizes faces in real-time, providing a convenient solution for attendance management. The project not only highlights the technical prowess of using OpenCV and TensorFlow but also emphasizes the potential of AI to enhance operational efficiency in various domains.

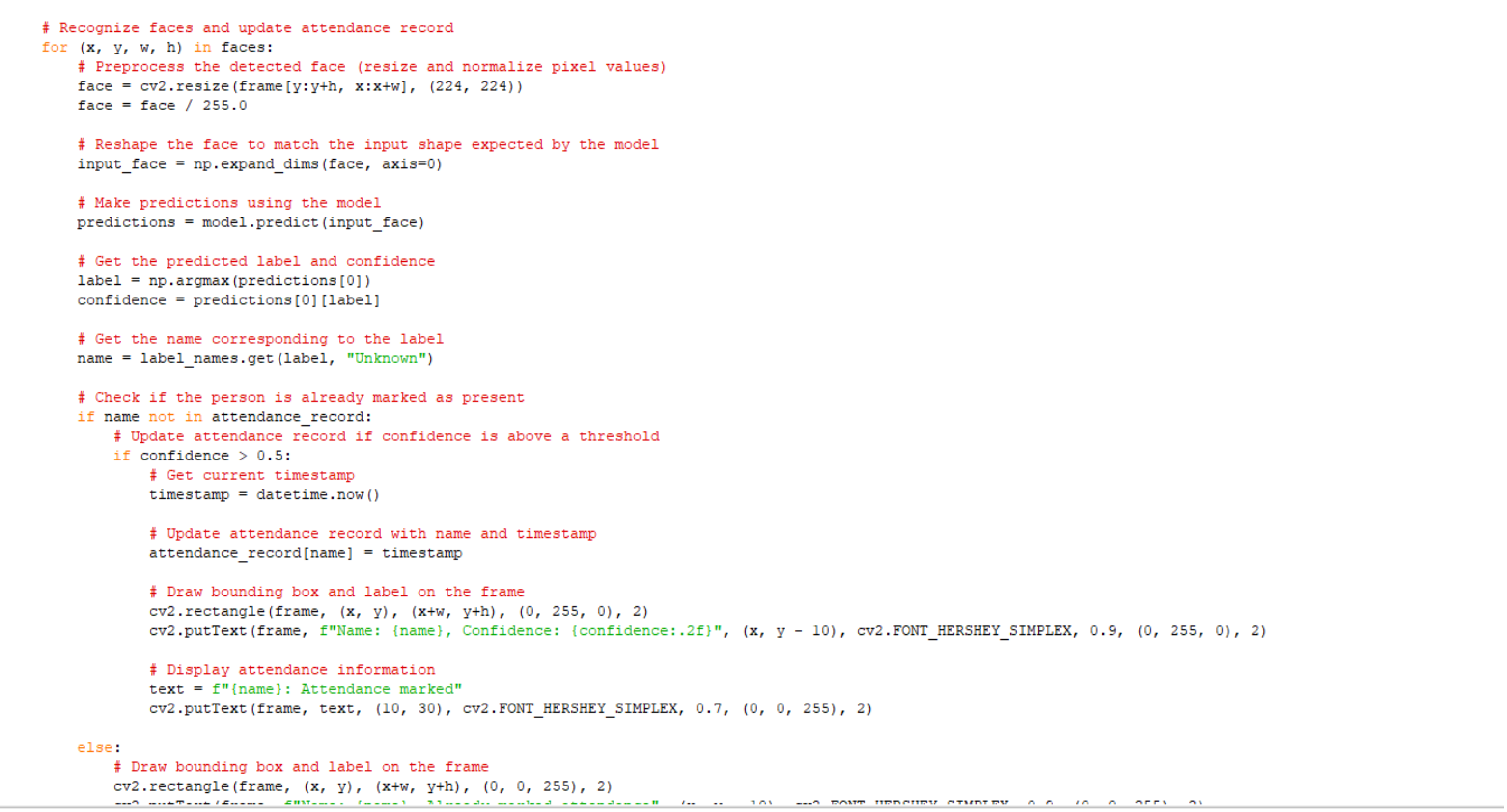
Future Enhancements

1. **Database Integration**: Integrate with a more sophisticated database system for improved data management and querying.
2. **Multiple Camera Support**: Extend the system to support multiple cameras simultaneously for larger venues.
3. **Online Mode**: Develop an online mode to facilitate remote attendance tracking.
4. **Real-time Analytics**: Provide real-time analytics on attendance patterns and trends.
5. **Enhanced Security**: Implement additional security measures to prevent unauthorized access and misuse.

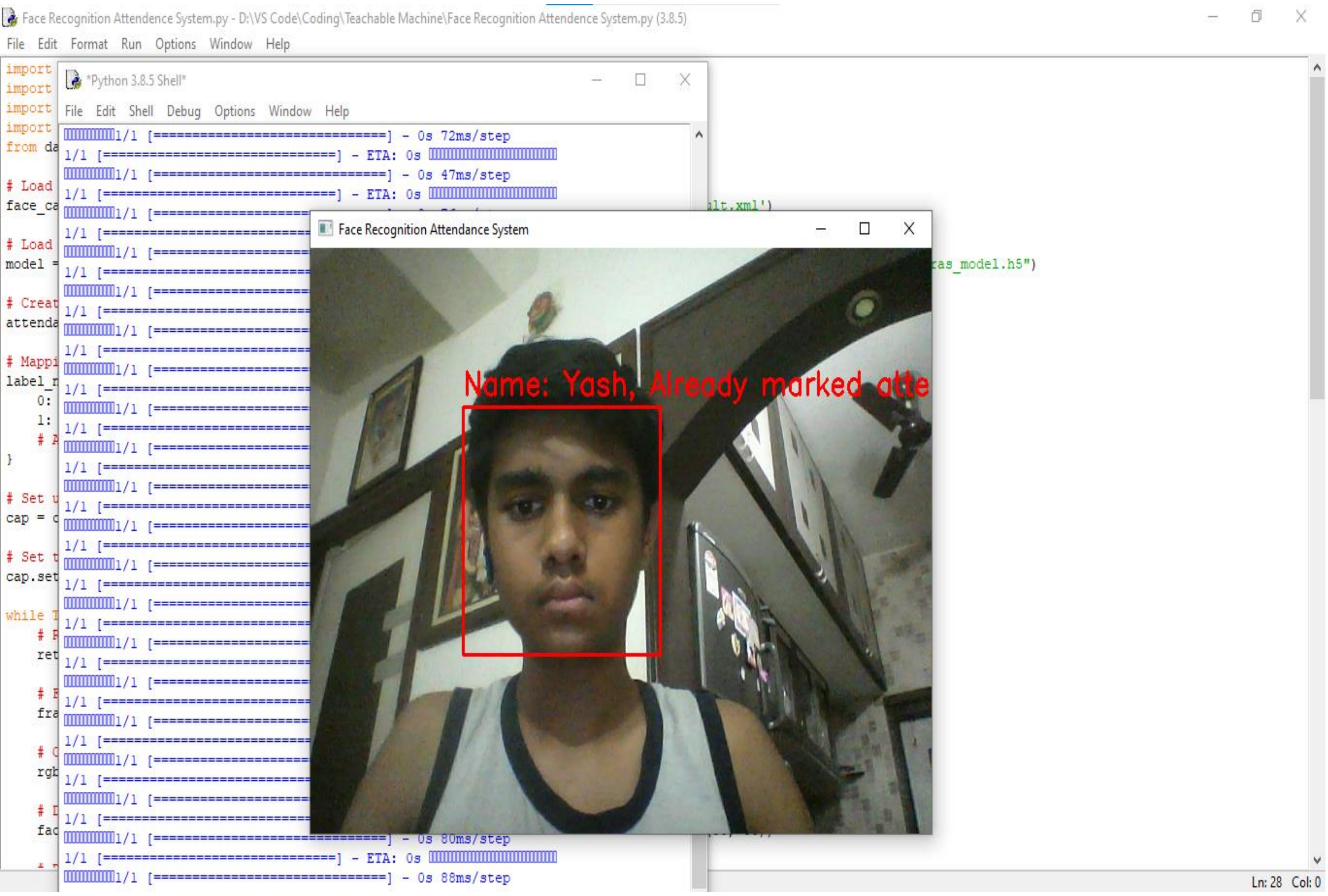
Screenshots



```python
import cv2
import numpy as np
import tensorflow as tf
import csv
from datetime import datetime

# Load the pre-trained face cascade classifier
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Load the exported TensorFlow.js model from Teachable Machine
model = tf.keras.models.load_model(r"D:\VS Code\Coding\Teachable Machine\face recogintion tensorflow converted_keras\keras_model.h5")

# Create attendance record (replace with your own database or record)
attendance_record = {}

# Mapping of label numbers to names
label_names = {
    0: 'Yash',
    1: 'Person 2',
    # Add more labels and names as needed
}

# Set up video capture
cap = cv2.VideoCapture(0)

# Set the frame rate to 60 fps
cap.set(cv2.CAP_PROP_FPS, 60)

while True:
    # Read video frame
    ret, frame = cap.read()

    # Flip the frame horizontally (mirror effect)
    frame = cv2.flip(frame, 1)

    # Convert frame to RGB (Teachable Machine model expects RGB images)
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

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

    # Recognize faces and update attendance record
```



```python
    # Recognize faces and update attendance record
    for (x, y, w, h) in faces:
        # Preprocess the detected face (resize and normalize pixel values)
        face = cv2.resize(frame[y:y+h, x:x+w], (224, 224))
        face = face / 255.0

        # Reshape the face to match the input shape expected by the model
        input_face = np.expand_dims(face, axis=0)

        # Make predictions using the model
        predictions = model.predict(input_face)

        # Get the predicted label and confidence
        label = np.argmax(predictions[0])
        confidence = predictions[0][label]

        # Get the name corresponding to the label
        name = label_names.get(label, "Unknown")

        # Check if the person is already marked as present
        if name not in attendance_record:
            # Update attendance record if confidence is above a threshold
            if confidence > 0.5:
                # Get current timestamp
                timestamp = datetime.now()

                # Update attendance record with name and timestamp
                attendance_record[name] = timestamp

                # Draw bounding box and label on the frame
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                cv2.putText(frame, f"Name: {name}, Confidence: {confidence:.2f}", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

                # Display attendance information
                text = f"{name}: Attendance marked"
                cv2.putText(frame, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        else:
            # Draw bounding box and label on the frame
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
            cv2.putText(frame, f"Name: {name}, Already marked attendance", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)
```


Face Recognition Attendence System.py - D:\VS Code\Coding\Teachable Machine\Face Recognition Attendence System.py (3.8.5)
File Edit Format Run Options Window Help
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Face Recognition Attendance System
Name: Yash, Already marked atte
Ln: 28 Col: 0

```
import
import
import
import
from da

# Load
face_ca                                                               alt.xml')

# Load
model =                                                               ras_model.h5")

# Creat
attenda

# Mappi
label_n
    0:
    1:
    # A
}

# Set u
cap = c

# Set t
cap.set

while T
    # R
    ret

    # E
    fra

    # C
    rgb

    # D
    fac
```

```
▯▯▯▯▯▯▯▯▯▯▯1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - ETA: 0s ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - ETA: 0s ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [====================
1/1 [==========================
▯▯▯▯▯▯▯▯▯▯▯1/1 [==============================] - 0s 80ms/step
1/1 [==============================] - ETA: 0s ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯1/1 [==============================] - 0s 88ms/step
```

Acknowledgments

I would like to express our gratitude to OpenCV and TensorFlow for providing the tools and resources necessary for the development of this project. Additionally, I appreciate the support from the open-source community and the creators of Teachable Machine for their contributions to the field of AI and machine learning.

References

- OpenCV Documentation: https://docs.opencv.org/
- TensorFlow Documentation: https://www.tensorflow.org/
- Teachable Machine: https://teachablemachine.withgoogle.com/

By: Yash Chauhan

Form: 10th Raman