Face Recognition with Emotion Detection

—Tannu Kumari

1. Overview

- Implement face recognition using the face_recognition library.
- Integrate emotion detection using a pre-trained model.
- Incorporate anti-spoofing measures with a pre-trained anti-spoofing model.
- Logged data into excel .

2. Libraries Used

Face Recognition Project used these libraries.

1. cv2 is used for capturing video, image processing, and basic computer vision tasks.
2. face_recognition is specifically designed for face-related tasks, such as face detection and recognition.
3. os helps in managing files and directories, crucial for loading models and organizing data.
4. numpy is used for efficient numerical operations, particularly helpful for handling image data.
5. keras.models.model_from_json is part of Keras and is used for loading neural network models from JSON files.
6. pandas is used for creating and manipulating DataFrames, which is useful for organizing and storing data.

3. Project Creation

This project involves the Recognition of face with known folder(known_image) and unknown folders (unknown_faces) in which it store no.of images of every unknown person in unknown folders by creating a unique folder for each unknown one and by that it recognize the unknown person with no spoofing. Each recognized face is assigned a unique identifier and stored data into XSLX file.

The following project structure outline is here :

```
FaceRecognitionProject/
│
├── known_image/
│   ├── person1/
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   ├── ...
│   ├── person2/
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   ├── ...
│   ├── ...
│
├── unknown_faces/
│   ├── unknown_1/
│   │   ├── unknown1.jpg
│   │   ├── unknown1.jpg
│   │   ├── ...
│   ├── unknown_2/
│   │   ├── unknown2.jpg
│   │   ├── unknown2.jpg
│   │   ├── ...
│   ├── ...
│
├── antispoofing_models/
│   ├── antispoofing_model.json
│   └── antispoofing_model.h5
│
├── models/
│   └── haarcascade_frontalface_default.xml
```

```

├── face_recog.py
├── facialemotionmodel.json
├── facialemotionmodel.h5
├── output_data.xlsx
```

4. Application

The Face Recognition project described aims to create a system capable of recognizing faces, detecting emotions, and implementing anti-spoofing measures. Here's how the application might be used:

- **Face Recognition:**
  - Utilizes the `face_recognition` library for face detection and recognition.
  - Compares unknown faces with known faces to identify individuals.
  - Assigns a unique identifier to each recognized face.
- **Emotion Detection:**
  - Implements emotion detection using a pre-trained model (`facialemotionmodel.json` and `facialemotionmodel.h5`).
  - Extracts face regions from recognized faces.
  - Predicts and labels emotions (angry, disgust, fear, happy, neutral, sad, surprise) in real-time.
- **Anti-Spoofing Measures**:
  - Utilizes an anti-spoofing model stored in the `antispoofing_models/` directory.
  - Captures faces in real-time using the webcam (`cv2.VideoCapture`).
  - Implements anti-spoofing measures to distinguish real faces from spoofed faces.
  - Labels each face as 'spoof' or 'real' based on the anti-spoofing model predictions.
- **Data Storage:**
  - Stores information for each recognized face into an Excel file (`output_data.xlsx`).

        - Includes the person's name (if known), detected emotion, and spoofing label ('spoof' or 'real').
- **Execution:**
    - Run the `project_code.py` script to initiate face recognition, emotion detection, and anti-spoofing processes.
    - View real-time results in the application displaying video feed with recognition outcomes.
    - Check the `output_data.xlsx` file for recorded data on recognized faces.
- **Further Improvements:**
    - Add a graphical user interface (UI) for a more user-friendly experience.
    - Optimize code for improved performance and scalability, especially with larger datasets.
    - Implement additional security measures based on specific use-case requirements.

5. Source Code

```
import cv2

import face_recognition

import os

import numpy as np

from keras.models import model_from_json

import pandas as pd



# Load emotion detection model

json_file = open("facialemotionmodel.json", "r")

model_json = json_file.read()

json_file.close()

emotion_model = model_from_json(model_json)

emotion_model.load_weights("facialemotionmodel.h5")
```

```
# Define emotion labels
emotion_labels = {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'happy', 4: 'neutral', 5: 'sad', 6: 'surprise'}

# Load Anti-Spoofing Model
json_file_spoof = open('antispoofing_models/antispoofing_model.json', 'r')
loaded_model_json_spoof = json_file_spoof.read()
json_file_spoof.close()
model_spoof = model_from_json(loaded_model_json_spoof)
model_spoof.load_weights('antispoofing_models/antispoofing_model.h5')
print("Anti-Spoofing Model loaded from disk")

# Load Face Detection Model
face_cascade = cv2.CascadeClassifier("models/haarcascade_frontalface_default.xml")



# Function to preprocess the image for emotion detection
def preprocess_image(image):
    resized_image = cv2.resize(image, (48, 48))
    gray_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
    normalized_image = gray_image / 255.0
    return normalized_image.reshape(1, 48, 48, 1)

faces_dir = r"C:\Users\HP\Desktop\Face_Antispoofing_System-main\known_image"
unknown_faces_dir =
r"C:\Users\HP\Desktop\Face_Antispoofing_System-main\unknown_faces"

known_face_encodings = []
known_face_names = []
```

```
for filename in os.listdir(faces_dir):
    if filename.endswith(".npz"):
        path = os.path.join(faces_dir, filename)
        data = np.load(path, allow_pickle=True)
        known_face_encodings.append(data['encoding'])
        known_face_names.append(data['name'])

known_face_encodings_dict = {tuple(encoding): name for encoding, name in
zip(known_face_encodings, known_face_names)}

confidence_threshold = 0.6

cap = cv2.VideoCapture(0)

unknown_person_dict = {}

# Create an empty DataFrame to store data
columns = ['Name', 'Emotion', 'Spoof']
data_df = pd.DataFrame(columns=columns)

# Create a dictionary to keep track of added faces
added_faces = {}

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

    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)

    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
```

```
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

    # Extract face region for emotion detection
    face_region = frame[top:bottom, left:right]
    preprocessed_face = preprocess_image(face_region)

    # Predict emotion
    emotion_prediction = emotion_model.predict(preprocessed_face)
    emotion_label = emotion_labels[np.argmax(emotion_prediction)]

    # Display emotion label on the frame
    cv2.putText(frame, f'Emotion: {emotion_label}', (left + 6, bottom + 20),
cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 255), 1)

    matches = face_recognition.compare_faces(known_face_encodings, face_encoding,
tolerance=confidence_threshold)

    if any(matches):
      first_match_index = matches.index(True)
      name = known_face_names[first_match_index]
      cv2.putText(frame, f"Known: {name}", (left + 6, bottom - 6),
cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 255), 1)
    else:
      face_encoding_tuple = tuple(face_encoding)

      found_match = False
      for known_encoding, identifier in unknown_person_dict.items():
        score = face_recognition.face_distance(np.array([known_encoding]),
np.array(face_encoding_tuple))[0]
```

```
                    if score < confidence_threshold:
                        identifier = unknown_person_dict[known_encoding]
                        found_match = True
                        break

                if found_match:
                    name = f"Unknown_{identifier}"
                else:
                    identifier = len(unknown_person_dict) + 1
                    unknown_person_dict[face_encoding_tuple] = identifier
                    name = f"Unknown_{identifier}"

                    unknown_person_folder = os.path.join(unknown_faces_dir,
f"unknown_{identifier}")
                    os.makedirs(unknown_person_folder, exist_ok=True)

                cv2.putText(frame, f"Unknown: {name}", (left + 6, bottom - 6),
cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 255), 1)

                img_counter = len(os.listdir(unknown_person_folder)) + 1
                if img_counter <= 5:
                    person_img = frame[top:bottom, left:right]
                    person_filename = f"unknown_{identifier}_{img_counter}.jpg"
                    person_path = os.path.join(unknown_person_folder, person_filename)

                    cv2.imwrite(person_path, person_img)

            # Perform Anti-Spoofing
            face = frame[top:bottom, left:right]
            resized_face = cv2.resize(face, (160, 160))
```

```
        resized_face = resized_face.astype("float") / 255.0
        resized_face = np.expand_dims(resized_face, axis=0)
        preds_spoof = model_spoof.predict(resized_face)[0]

        if preds_spoof > 0.5:
          label_spoof = 'spoof'
          cv2.putText(frame, f'Anti-Spoofing: {label_spoof}', (left + 6, bottom + 40),
                cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255), 1)
        else:
          label_spoof = 'real'
          cv2.putText(frame, f'Anti-Spoofing: {label_spoof}', (left + 6, bottom + 40),
                cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 255, 0), 1)

          # Update DataFrame with current data
          data_df = pd.concat([data_df, pd.DataFrame({'Name': [name], 'Emotion':
[emotion_label], 'Spoof': [label_spoof]})], ignore_index=True)


      cv2.imshow("Face Recognition with Emotion Detection and Anti-Spoofing", frame)

      if cv2.waitKey(1) & 0xFF == ord('q'):
        # Save the DataFrame to an Excel file before exiting
        excel_filename = 'output_data.xlsx'
        data_df.to_excel(excel_filename, index=False)
        Break
```

6. References

https://github.com/tannukumari742/Facial_recognition