Documentation

Crowd Monitoring and Analyses

Mustafa Tariq (223124219)

Abdul Ahad (223735551)

Project Overview
This project focuses on developing a face detection and characteristics extraction model using deep learning techniques. The model is capable of processing video frames, detecting faces, extracting facial landmarks, and computing face embeddings for recognition purposes. The project integrates YOLOv3 for face detection and Dlib for facial landmark prediction and embedding extraction.

Code Overview
The code is structured to process a video file, detect faces in each frame, extract facial characteristics, and generate a JSON-like object containing the details of detected faces. The key components of the code are:

1. **Loading YOLO Model:**
    - The load_yolo_model function initializes the YOLOv3 model with the given weights and configuration files. This model is used for face detection.

      ```
      load_yolo_model(weights_path, config_path)
      ```

        - **Purpose**: Loads the YOLO model using the specified weights and configuration files.
        - **Parameters**:
            - `weights_path`: Path to the YOLO weights file.
            - `config_path`: Path to the YOLO configuration file.
        - **Returns**: The YOLO network and output layers.

2. **Face Detection:**
    - The detect_faces function processes each frame to detect faces using the YOLOv3 model. It returns the coordinates of detected faces along with the confidence scores for each detection. A face is considered detected if the confidence score exceeds the 50% threshold.

      ```
      detect_faces(img, net, output_layers, confidence_threshold=0.5)
      ```

        - **Purpose**: Detects faces in an image using the YOLO model.

- **Parameters**:
  - `img`: Image in which faces are to be detected.
  - `net`: Loaded YOLO network.
  - `output_layers`: Layers from the YOLO model.
  - `confidence_threshold`: Minimum confidence to consider a detection valid.

- **Returns**: Processed image with bounding boxes, list of detected face coordinates, and their confidence scores.

3. **Facial Characteristics Extraction:**
   - The extract_face_characteristics function extracts facial landmarks and computes the embedding vector for each detected face using Dlib. The landmarks are also highlighted in the output video.

     `extract_face_characteristics(img, faces, shape_predictor, face_rec_model)`

     - **Purpose**: Extracts facial landmarks and embeddings for detected faces.
     - **Parameters:**
       - `img`: Image containing detected faces.
       - `faces`: List of face coordinates.
       - `shape_predictor`: Dlib shape predictor for facial landmarks.
       - `face_rec_model`: Dlib face recognition model.
     - **Returns**: List of face characteristics including bounding box, landmarks, and embeddings.

4. **Processing Video Frames:**
   - The process_video function processes the video file frame by frame, detects faces, extracts characteristics, and stores the results in a JSON-like object. The results are printed for each frame and can also be saved to a JSON file.

     `process_video(video_path, output_video_path, net, output_layers, shape_predictor, face_rec_model)`

     - **Purpose**: Processes a video to detect faces in each frame, extracts characteristics, and outputs a video and JSON data.
     - **Parameters:**
       - `video_path`: Path to the input video.
       - `output_video_path`: Path to save the output video.
       - `net`, `output_layers`, `shape_predictor`, `face_rec_model`: Preloaded models.
     - **Returns**: JSON-like object with details of detected faces per frame.

5. **Main Function:**
    - The main function serves as the entry point of the code, loading the models and calling the video processing function. The output video is saved with detected faces highlighted, and the results are saved to a JSON file.

      `main()`

        - **Purpose**: Entry point for the script to process the video using the above functions.

Confidence Scores and Accuracy Assessment

In this project, the accuracy of face detection is assessed using the confidence scores provided by the YOLO model. The model is configured with a confidence threshold of 50%, meaning that only faces detected with a confidence score above this threshold are considered valid detections. While this method does not provide a direct accuracy metric, it ensures that only faces detected with reasonable certainty are included in the results.

The confidence scores are included in the JSON-like object generated for each frame, providing a measure of the reliability of each detected face.

Weekly Progress

Week 1:

- **Activity:** Completed the implementation of the face detection model using YOLOv3.
- **Challenges:** Encountered issues with setting up the YOLO environment, including installation of necessary libraries and configuration of the model weights and configuration files.
- **Outcome:** Successfully set up the environment and completed the face detection module, capable of detecting faces in images and video frames.

Week 2:

- **Activity:** Developed the facial characteristics and shape predictor models using Dlib.
- **Challenges:** Faced difficulties with the installation of Dlib and its dependencies, requiring manual compilation and troubleshooting of environment issues.
- **Outcome:** Completed the facial landmark detection and embedding extraction modules, enabling detailed analysis of detected faces.

Week 3:

- **Activity:** Integrated the face detection and facial characteristics models into a video processing pipeline.
- **Challenges:** Encountered issues with processing large video files, managing memory, and ensuring the performance of the integrated system.
- **Outcome:** Successfully integrated a video module that processes each frame, detects faces, extracts characteristics, and generates a JSON-like object with detailed information about frames, detected faces, including bounding boxes, landmarks, embeddings, and confidence scores.

Installation and Environment Setup

Throughout the project, significant effort was invested in setting up the development environment, which involved:

- Installing OpenCV for image and video processing tasks.
- Setting up the YOLOv3 model, including downloading and configuring the necessary weights and configuration files.
- Installing Dlib, which required manual compilation due to compatibility issues with the existing system setup.
- Ensuring all dependencies were correctly installed and configured, allowing the models to function smoothly together in an integrated pipeline.

The environment setup was a critical step in the project, as it allowed for the successful implementation and integration of advanced computer vision techniques.