ATTENDANCE MANAGEMENT SYSTEM

using Cam

A project report submitted to ICT Academy of Kerala

in partial fulfillment of the requirements

for the certification of

INTERNSHIP PROJECT

IN

DATA SCIENCE & ANALYTICS

Submitted by

AKSHAY A R

ICT ACADEMY OF KERALA

THIRUVANANTHAPURAM, KERALA, INDIA

June 2024

INTRODUCTION

This project is designed as an automated attendance tracking system that leverages face recognition technology to accurately record and manage attendance. Built with deep learning and computer vision techniques, the system aims to streamline attendance processes, reducing manual effort and increasing reliability.

Project Overview

The attendance tracking system captures real-time images of users through a webcam and processes these images using a Convolutional Neural Network (CNN). Each time the system detects a bounding box around a user's face, it captures 100 images to build an extensive image dataset for accurate recognition. After data collection, the model is trained to identify users, creating an automated and efficient way to manage attendance records.

Key Technologies

1. **Flask**: This lightweight web framework is used to run the application, providing a user-friendly interface and managing backend tasks.
2. **OpenCV**: For image capture and processing, OpenCV detects faces through the webcam feed, ensuring accurate data collection.
3. **SQLite**: A simple, efficient SQLite database is implemented to store and manage attendance records, ensuring that data is organized and easily accessible.
4. **Python (TensorFlow/Keras)**: The CNN model is built using TensorFlow and Keras libraries, including two additional Conv2D and MaxPooling layers to improve accuracy. The system displays graphs for model accuracy and loss, providing insight into training performance.

Technology Stack

- **Programming Languages**: Python, SQL, HTML, CSS
- **Libraries/Frameworks**: Flask, OpenCV, Keras (for CNN), MySQL
- **Tools**: VS Code
- **Database Management**: MySQL

Workflow

1. **Image Capture:** The system activates the webcam and uses OpenCV to detect faces. Once a face is detected, it captures and stores 100 images per user.
2. **Data Preprocessing:** Images are preprocessed to optimize the training process.
3. **Model Training and Evaluation:** A CNN model with enhanced layers is trained on the captured dataset, achieving high accuracy in identifying users.
4. **Prediction and Attendance Tracking:** Once the model is trained, it predicts user identities in real-time and logs attendance entries in the SQLite database.
5. **New User Registration:** The system allows for the addition of new users by capturing and adding their images, updating the dataset for continuous learning.

Application Highlights

- **Accuracy and Efficiency:** The model's added layers ensure high accuracy, while OpenCV and Flask make real-time image capture and user recognition efficient.
- **Simple Data Management:** SQLite provides a lightweight and easy-to-manage database solution.
- **Scalability:** The system supports new user additions, with capabilities to grow and accommodate multiple users.

Project Attachment

Github link: https://github.com/akshayar777/DSA2024

Detailed Modules and Functions

1. Frontend (app.py)

The frontend, built with Flask, serves as a bridge between users and the backend functions. Key features include:

- **User Authentication**: Secure login for admins or authorized personnel.
- **Start/Stop Attendance Capture**: Allows admins to initiate or terminate the attendance logging process.
- **Attendance Records Access**: Admins can view, filter, and download attendance logs.

2. Backend (image_capture.py)

This module handles the camera interface, real-time face detection, and image pre-processing for the CNN model. Key components include:

- **Image Capture**: Initiates camera access to capture images in real-time.
- **Face Detection and Pre-processing**:
    - Uses OpenCV to detect faces in each frame.
    - Crops, resizes, and normalizes images for CNN model compatibility.
- **Data Handling**: Prepares and formats captured images, saving them for attendance logging or model training.

3. Modeling and Prediction

- **Model Architecture**: A Convolutional Neural Network (CNN) model is used, designed with several convolutional, pooling, and fully connected layers.
- **Training and Testing**: The model is trained on a dataset of known individuals with labeled images, allowing it to learn and recognize unique facial features.
- **Prediction**: The trained CNN model predicts the identity of individuals from the processed frames.
- **Output and Probability**: Returns an individual's ID with an associated probability score to ensure accuracy.

4. Testing

- **Accuracy Testing**: The model undergoes various testing phases, including real-time validation, to ensure high recognition accuracy.
- **Threshold Adjustment**: Probability thresholds are fine-tuned to reduce false positives and negatives.

- **Performance Metrics**: Evaluation metrics such as precision, recall, and F1 score are tracked.

5. Database Management (DBMS) with MySQL

- **User Authentication**: Stores admin credentials, securing access to the system.
- **Attendance Records**:
    - **Table Design**: Each table entry includes fields for user ID, timestamp, and attendance status.
    - **CRUD Operations**: Attendance records can be created, updated, and retrieved as needed.
- **Attendance Query**: SQL queries fetch, filter, and organize attendance data based on dates, users, or other criteria

System Flow and Workflow

1. **User Login**: Authorized personnel log into the system via `app.py`.
2. **Attendance Capture**: Upon starting attendance capture:
    - image_capture.py activates the camera.
    - Captures images in real-time.
    - Detects faces in each frame, preprocessing them.
3. **Prediction and Logging**:
    - Preprocessed images are fed into the CNN model.
    - If a match is found, the individual's ID is logged.
    - Attendance data is stored in the MySQL database, associating the user ID with a timestamp.
4. **Viewing Records**: Admins can query attendance records from the database, filtering by date, ID, or other criteria.

Testing and Evaluation

1. **Model Evaluation**: The CNN model was evaluated on a test dataset.
    - Achieved an accuracy of approximately **94%** (after final testing).
    - Adjustments to architecture and hyperparameters based on testing results.

2. **System Testing**: Real-world scenarios were tested to simulate attendance logging for multiple individuals at varying times.
    - **Accuracy**: Confirmed attendance logging accuracy for a sample group with minimal errors.
    - **Prediction Time**: System was able to predict within .18 - .28 seconds of detection.
    - **Reliability**: The system could handle multiple faces in a frame.

Results

The Attendance Management System successfully automated the process of attendance logging. It was able to:

- Detect and recognize faces in real-time.
- Log attendance with a high degree of accuracy.
- Provide secure access and management of attendance data.
- Store records in a structured database for easy access and analysis.

Challenges and Solutions

1. **Low Lighting and Image Quality**:
    - **Solution**: Pre-processing techniques and image augmentation were implemented to improve recognition in low-light settings.
2. **Model Accuracy**:
    - **Solution**: Used a larger dataset with diverse images, fine-tuned model parameters, and added more layers to improve performance.
3. **Database Efficiency**:
    - **Solution**: Indexing and query optimization techniques were applied to manage large attendance data sets efficiently.

Future Enhancements

1. **Mobile Integration**: Develop a mobile application for more accessible attendance logging and viewing.
2. **Multi-Camera Support**: Enable multi-camera functionality to monitor larger spaces.
3. **Cloud Integration**: Store data in a cloud-based system for remote access and backup.

Image Capture Process

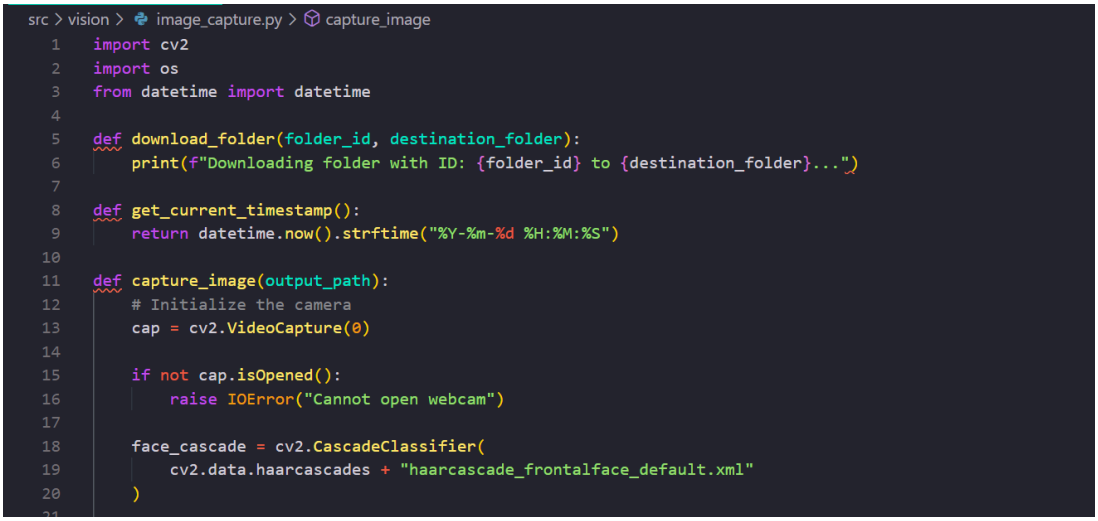
The Image Capture module is fundamental to the Attendance Management System, responsible for real-time face detection and image storage, ensuring accurate attendance tracking and face recognition. This module activates the webcam using OpenCV, a popular computer vision library, and continuously monitors the video stream for facial detection. Once a face is detected, the system captures and saves 100 images of the individual to build a robust dataset for model training and recognition.

Step-by-Step Process:

- **Face Detection:**
    - The camera initializes via the backend, and OpenCV begins a live video stream feed.
    - Using OpenCV's `CascadeClassifier` or `DNN`-based face detectors, the system scans each frame for faces.
    - When a face is identified, a rectangle is drawn around the detected face to visually confirm detection in the user interface.
- **Image Capture and Temporary Storage:**
    - The capture_image function is triggered within the frontend application (`app.py`). This function initiates face capture through the live stream, where each frame containing the detected face is saved temporarily.
    - To avoid redundancy, images are captured sequentially once the face is detected within the rectangle bounds.
    - The captured images are saved in the 'temp' folder as 'temp.jpg' by the save_image function, supporting the initial phase of training and allowing quick inspection of each image captured.

- **Sequential Capture and Image Storage:**
    - To build a reliable dataset for each individual, the system captures up to 100 sequential images per user in real time. This approach provides diverse data by capturing slight variations in angle, lighting, and expression, which improve model performance in recognizing faces under different conditions.

- Captured images are stored permanently in the ‘captured images’ folder. This organized folder structure assists in consistent data retrieval, model training, and ongoing recognition processes.



```
src > vision > image_capture.py > capture_image
import cv2
import os
from datetime import datetime

def download_folder(folder_id, destination_folder):
    print(f"Downloading folder with ID: {folder_id} to {destination_folder}...")

def get_current_timestamp():
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")

def capture_image(output_path):
    # Initialize the camera
    cap = cv2.VideoCapture(0)

    if not cap.isOpened():
        raise IOError("Cannot open webcam")

    face_cascade = cv2.CascadeClassifier(
        cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
    )
```

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

        if ret:
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

            faces = face_cascade.detectMultiScale(gray, 1.3, 5)

            for x, y, w, h in faces:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

            cv2.imshow("Face Detection", frame)

            if len(faces) > 0:
                cv2.imwrite(output_path, frame)
                print(f"Image saved at {output_path}")
                break

        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
```

- **Prediction and Verification:**
  - Once images are stored, they are used for real-time recognition and matched against an employee database. Each detected face undergoes prediction through the trained model to identify the correct individual.
  - The Database Verification* phase matches the identified user with database records, confirming the correct employee ID. Attendance details, including a timestamp, are then securely stored in the database, enabling administrators to track attendance accurately.

- **Logging Attendance:**
    - After successful verification, attendance data is logged automatically. Each record includes the employee ID, timestamp, and attendance status, securely stored within the database for future access and analysis.

Contributions:

- **Akshay A R :** capture_image function creation, OpenCV, connect with livestream, updating infer.py for prediction time
- **Karthika T K :** save_image function, adding timestamp, connect with livestream

CONCLUSION

The Image Capture module combines real-time face detection, sequential image capture, and database verification to automate the attendance management process efficiently. By capturing multiple images per user and storing them systematically, the system ensures accurate face recognition. This rigorous approach aids in consistent employee identification, even in varying conditions. The database-backed logging system provides reliable records of attendance with timestamps, enhancing data accuracy and accessibility.

In conclusion, the Image Capture process plays a pivotal role in making the Attendance Management System both reliable and user-friendly. It reduces the need for manual attendance marking, increases operational efficiency, and ensures secure, real-time attendance tracking. This automated solution demonstrates the power of computer vision and AI in solving practical challenges in attendance management.

The project was a comprehensive solution to attendance management using a CNN-based face recognition system. Through real-time detection, secure storage, and reliable access, the Attendance Management System proved to be an effective tool for accurate and efficient attendance tracking. This project demonstrates the effective use of computer vision and deep learning techniques to solve real-world problems.