Smart Attendance System

In Fulfillment of Semester Project

Deep Neural Networks

Instructor: Dr. Ali Imran

Project Report

Written By: Alishba Bacha 2022088
Aiman Darakhshan 2022076
Reiaan Mazhar 2022502
Wardah Haya 2022660

Date: December 26, 2024

1. Dataset

The dataset used for the project consists of images of students in the class, which were uploaded to Roboflow. The dataset was organized into two primary folders: train and valid. Each folder contained subfolders for images and corresponding labels, formatted to match the YOLO annotation style. These labels contain the coordinates of each student's face in the images, which are used for object detection during the model training process. We have included a YAML file with detailed configurations for the model, including paths to the images and annotations. This YAML file is used to ensure the model can correctly interpret the dataset during training and prediction.

2. Model Training

The Smart Attendance System model training was carried out using YOLOv8, which was installed and configured with the necessary dependencies. First, the data set was extracted from a ZIP file containing student images and labels. These images were organized and prepared for YOLO by separating them into appropriate folders for images and labels, using the prepare yolo dataset function.

To facilitate the model's training process, the train yolomodel function was used, where the YOLOv8 model was initialized and trained on the dataset. The model was trained for 250 epochs with images resized to 640px, allowing it to learn from the dataset. During the training process, the model was fine-tuned to detect students' faces from their images, assigning each student a unique registration number as a class label.

Once the model training was complete, the trained YOLO model was used for detection. The detect and identify person function processed group images, identifying students based on their unique registration numbers. This detection was enhanced by using class labels corresponding to each student's registration number, which were provided in a list. The model was supposed to predict the presence of the students in the image, with the results saved for further processing.

Then the attendance was to be marked for each detected student by recording the detected registration numbers, the current date, and the class name in a CSV file, as managed by the mark attendance function.

3. Backend

The backend of the Smart Attendance System was developed using Flask, leveraging SQLite for database management. The Flask app handles both user authentication and attendance marking functionality. It integrates YOLO for student detection and records attendance based on the results.

Flask App Setup:

The app uses Flask to handle routing and user requests, and Flask-Login for user authentication.

The app is secured with a secret key and has a login manager to handle user sessions and redirects.

An uploads folder is created to store photos uploaded by users (teachers).

Database:

The system uses SQLite to manage user data and attendance records. It initializes the database with two main tables: o users table to store the usernames, passwords , and user roles. o attendance table to log attendance records, including the date, day, class name, and student registration number. SQL queries are executed within the app.py file to interact with the database, such as creating new users, checking login credentials, and marking attendance.

User Authentication:

Signup: New users can sign up by providing a username, password, and role. The data is securely stored in the database.

Login: Users can log in by entering their credentials. Teachers are redirected to a dashboard where they can upload group photos for attendance marking, while students can view their attendance records.

Logout: Users can log out, which ends their session and redirects them back to the home page.

Attendance Functionality:

Teachers can upload a group photo from the teacher dashboard, where YOLO is used to detect students in the image. The model identifies the students and returns their registration numbers.

The detected students' attendance is then logged in the attendance table in the database.

The system supports attendance tracking for students, who can view their attendance records on the student dashboard.

YOLO Integration:

YOLO is used for student detection. The trained YOLO model is loaded into the backend, where it processes the uploaded group photo to identify students and match them to their records in the database.

Upon detecting students in the image, their registration numbers are extracted and used to mark attendance

4. Frontend

The frontend was designed using HTML, CSS, and JavaScript to provide an interactive user interface. The HTML files, such as login.html, signup.html, attendance.html, teacher.html, and others, were used to create the following functionality:

Login and Signup Pages:

Users could sign up as either students or teachers and log into the system using their credentials.

Teacher Dashboard:

Teachers could upload group photos of students for attendance marking. The page also allowed teachers to select the course and section they were handling.

Previous Attendance Page:

This page displayed a table with records of previous attendance, showing the date, course, and student attendance status.

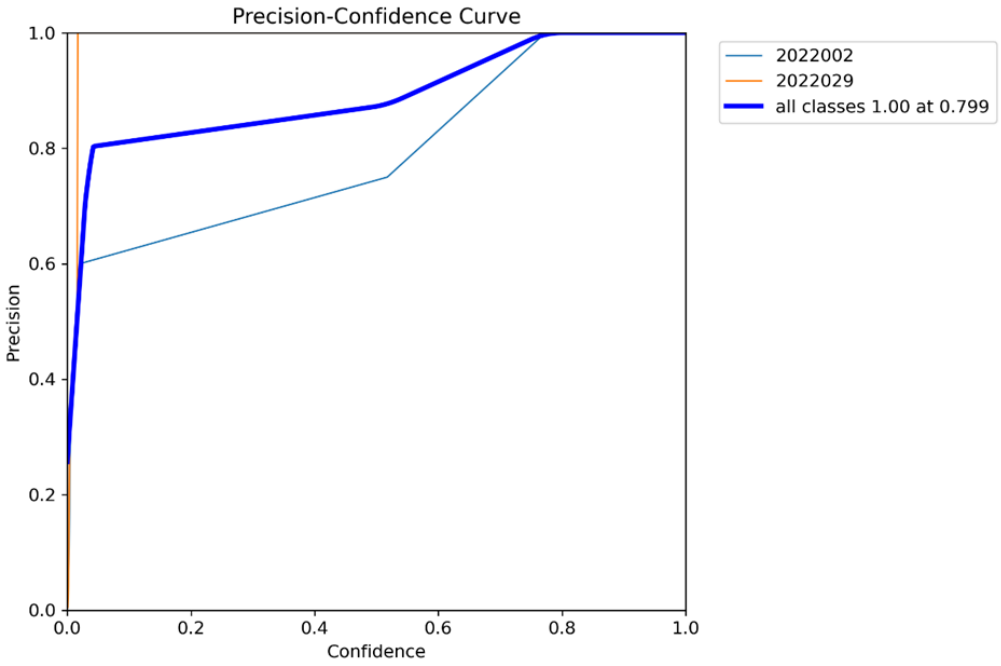
Logout Page:

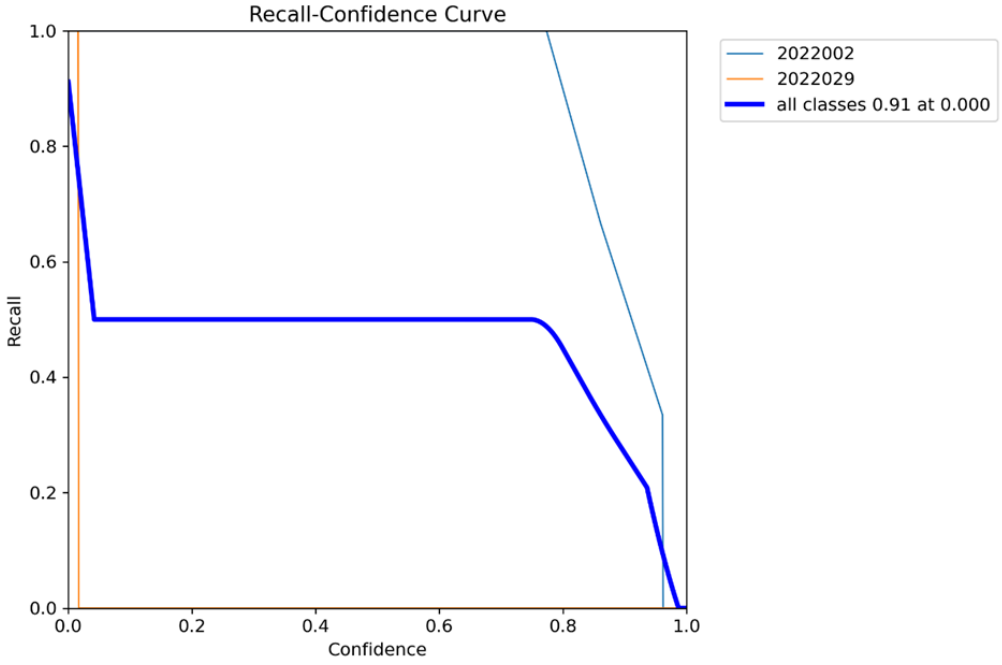
Teachers and students could log out of the system using a simple logout page with options to either return to the homepage or log in again.

The style.css file was used for styling the frontend, ensuring a clean and responsive layout. It defined the styles for the navigation bar, buttons, headers, forms, and other UI elements. Attached below is the picture of the frontend.

5. Results

Although training was successful, the model predictions were inaccurate, where only two student identifiers, “2022002” and “2022029,” were detected for multiple students as you can see in the graphs of precision and recall attached below.





6. Problems Faced and How We Resolved Them

1.Dataset Format Issues:

o Initially, the dataset was in different formats, causing compatibility issues with the YOLO model. This was resolved by ensuring that all images and labels were correctly formatted according to the YOLO annotation style and uploaded to Roboflow. A YAML file was also created to specify the dataset structure.

2.Training YOLOv5:

o YOLOv5 was attempted first but failed to train properly. We encountered issues related to model configuration. The solution was to switch to YOLOv8 and YOLOv11, which successfully trained the model, though they still produced inaccurate results.

3. Switching from Flutter to Flask:

o The initial approach was to build the frontend using Flutter, but it kept crashing. After troubleshooting, it was decided to switch to Flask for the backend, integrating the HTML, CSS, and JavaScript frontend with Flask for better stability and functionality.

4. Database Issues with PostgreSQL:

o We initially used PostgreSQL for the database, but encountered setup issues and compatibility problems. This was resolved by switching to SQLite and embedding the database logic directly in app.py using SQL queries.