AI Attendance System: Full Technical Documentation

Contents

AI Attendance System - Technical Documentation and Calculations

1) System Overview

This document describes the AI-based Attendance System, including:

- Architecture (frontend + backend)
- Database structure and API endpoints
- Face detection, embedding extraction, and cosine similarity matching
- Thresholding and blur filtering
- Attendance marking pipeline and calculations

2) High-Level Workflow

Student Registration:

1. Student creates an account with name, matric number, and password.
2. Student uploads multiple face images (front/left/right views or multiple samples).
3. System detects and crops faces from each image.
4. System extracts embeddings using a face recognition model.
5. Embeddings are normalized and saved for future matching.

Taking Attendance (Live Capture / Scan):

1. Lecturer starts a session for a course.
2. Webcam captures frames periodically.
3. Each frame is processed: face detection $\rightarrow$ cropping $\rightarrow$ embedding extraction.
4. Embeddings are matched against the gallery.
5. Matched student IDs are marked present for the active session.

2) Data Model Summary

Tables (simplified):

1. users: id, name, role, identifier, department
2. courses: id, course_code, course_title, lecturer_id, enrollment window
3. course_departments: course_id, department
4. enrollments: student_id, course_id
5. sessions: course_id, lecturer_id, status, start_time, end_time
6. attendance: session_id, student_id, timestamp, method, confidence
7. face_embeddings: student_id, view_type, embedding_path

3) Face Detection and Cropping

The system uses OpenCV YuNet (if available) or Haar cascade as fallback. A face bounding box is produced as (x1, y1, x2, y2). The system pads and crops:

```
x1' = max(0, x1 - pad)
y1' = max(0, y1 - pad)
x2' = min(W, x2 + pad)
y2' = min(H, y2 + pad)
```

Where:

- W, H are image width and height.
- pad is a fixed padding (e.g., 20 pixels).

4) Blur Filtering (Image Quality Gate)

Before extracting embeddings, a blur score is computed using the variance of Laplacian:

```
blur_score = Var( Laplacian( grayscale(face) ) )
```

Decision rule:

- If blur_score ¡ blur_threshold, the face is skipped.
- Current default blur_threshold for scanning is 80.

Notes:

- Higher blur_threshold means stricter image quality.
- A lower blur_threshold allows blurrier images.

5) Embedding Extraction and Normalization

Let v be the raw embedding vector from the model (dimension d, e.g. 128 or 512).

L2 norm:

$$\|v\|_2 = \sqrt{\sum_{i=1}^{d} v_i^2}$$

Normalized embedding:

$$v_{norm} = \frac{v}{\|v\|_2 + \epsilon}$$

where $\epsilon = 10^{-8}$ avoids division by zero.

The system uses this normalized vector for cosine similarity.

6) Cosine Similarity Matching

Given two normalized embeddings a and b, cosine similarity is:

$$\cos(\theta) = a \cdot b = \sum_{i=1}^{d} a_i b_i$$

Because a and b are normalized, $\|a\|_2 = \|b\|_2 = 1$.

So cosine similarity lies in $[-1, 1]$, but in face embeddings it typically ranges from about 0.2 to 0.95.

6) Gallery Construction and Vectorized Matching

Each student has multiple embeddings per view (front, left, right). The gallery matrix G is built as:

```
G = [ v1^T
      v2^T
      ...
      vN^T ]
```

where each row is a normalized embedding. The system stores metadata for each row:

```
(meta[i] = (student_id, view_type)).
```

For a query embedding q (normalized), cosine similarities are:

```
sims = G @ q
```

The best match is:

```
idx = argmax(sims)
(best_id, best_view, best_sim) = meta[idx], sims[idx]
```

7) Threshold Decision and Confidence

After obtaining *best_sim*, the system decides if the face is recognized.

```
Let threshold = T.
If best_sim >= T then Recognized else Unknown.
```

Typical threshold values:

- $T = 0.75$: more lenient (higher recall, more false positives)
- $T = 0.80$: balanced default
- $T = 0.85$: strict (lower false positives, but more false negatives)

The system may store confidence as:

$$confidence = best\_sim$$

8) Attendance Marking Logic

A session is opened by a lecturer. A student can be marked present only if:

- session is active,
- student is enrolled in the course,
- student has not already been marked present for that session.

If recognized:

1. Create attendance record (session_id, student_id, timestamp, method="face", confidence=best_sim)
2. Return success response to frontend.

9) Timing and Duplicate Control

To prevent duplicate marking:

- Each student is marked at most once per session.
- If a recognized face appears again, it is ignored.

Optionally, if the system allows re-marking after some minutes:

```
Let now = current time
If now - last_marked_time > cooldown then allow else block
```

10) Attendance Percentage Calculations

Define:

- A_s = number of sessions student attended
- S = total number of sessions held

Attendance percentage:

$$Attendance\% = \frac{A_s}{S} \times 100$$

For a course with multiple sessions:

$$Attendance\%_{course} = \frac{\text{Count of Present Records}}{\text{Total Sessions}} \times 100$$

If an entire class attendance rate is needed:

```
attendance_rate = (total_marked / (enrolled_students * total_sessions)) * 100
```

For session participation rate:

```
session_rate = (marked_count / enrolled_students) * 100
```

Example:

- Enrolled students = 30
- Total sessions = 4
- Total attendance records = 90

```
attendance_rate = (90 / (30*4)) * 100 = 75%
```

11) Error Sources and Practical Considerations

False Positives:

- Occur when threshold is too low or embeddings overlap.
- Mitigation: increase threshold, improve lighting, use more samples per student.

False Negatives:

- Occur when face angle differs, poor lighting, blur, occlusion.
- Mitigation: store multiple views, reduce blur threshold slightly, ask students to face camera.

12) Testing Plan (Not Yet Started)

Current status:

- Backend implemented (Flask, database, embedding and matching pipeline).
- Frontend implemented (registration, upload, live scanning UI).
- Integration complete: API endpoints connect frontend to backend.
- Testing and evaluation (accuracy metrics, speed benchmarks) not yet started.

Planned evaluation metrics:

- Recognition accuracy (TP, FP, FN)
- Precision, Recall, F1-score
- Average recognition latency (ms per face)
- Impact of threshold T on accuracy:

```
- Test T = 0.75, 0.80, 0.85
```

13) References (Suggested)

- OpenCV Documentation (Face Detection, Laplacian Blur)
- Cosine Similarity in Face Recognition Systems
- FaceNet: A Unified Embedding for Face Recognition
- ArcFace: Additive Angular Margin Loss for Deep Face Recognition