Gait-Based Employee Authentication System Technical Report

1. Introduction

This report presents a gait-based authentication system that uses smartphone inertial sensors to verify employee identities for secure workplace access. The system processes triaxial accelerometer and gyroscope data to authenticate employees within 1 second, ensuring minimal battery usage (less than 3% per 8-hour shift) and low bandwidth (less than 20 KB/s per user). It accommodates variable phone orientations and integrates with a mock door controller API. This document covers data preparation, feature choices, model selection, tuning, latency tests, and security considerations, including spoofing and lost phone scenarios. A prior issue with label range mismatches was resolved to support the 30 subjects in the UCI Human Activity Recognition dataset.

2. Data Preparation

2.1 Data Ingestion

The system uses the UCI Human Activity Recognition dataset, which includes triaxial accelerometer and gyroscope signals from 30 unique subjects. Each sample consists of 128 timesteps across 6 channels (3 accelerometer, 3 gyroscope) sampled at 50 Hz, representing a 2.56-second window. Data is loaded from text files into a tensor of shape (samples, 128, 6). Subject IDs, ranging from 1 to 30, serve as authentication labels.

2.2 Preprocessing

To handle phone orientation variability (portrait or landscape), the system normalizes data by dividing each channel by the Euclidean magnitude across axes. This preserves gait patterns while reducing orientation sensitivity. The dataset is pre-normalized to the range [-1, 1], eliminating the need for additional scaling. Subject IDs are mapped to 0-based indexing (1 to 30 becomes 0 to 29) to align with the model's loss function. Missing or noisy data is minimal in the curated dataset, but real-world systems would require outlier detection.

3. Feature Engineering

The system adopts a deep learning approach, feeding raw inertial data into a 1D Convolutional Neural Network (1D-CNN). This eliminates manual feature extraction, such as calculating mean, standard deviation, or spectral entropy, as used in classical methods. The 1D-CNN processes the input tensor (128 timesteps, 6 channels) to learn hierarchical temporal features, offering greater robustness to noise and orientation changes compared to handcrafted features, which may overlook subtle gait patterns.

4. Model Selection and Tuning

4.1 Model Architecture

A 1D-CNN was selected for its efficiency in processing time-series data. The architecture includes:
- Input Layer: Accepts tensors of shape (128, 6).
- Conv1D Layers: Two layers with 64 and 128 filters, kernel size 3, and ReLU activation to extract temporal patterns.
- MaxPooling1D Layers: Pool size 2 to reduce dimensionality.
- Dense Layers: A 256-unit layer with ReLU activation, followed by a 30-unit softmax layer for classifying 30 subjects.
- Dropout: 0.5 rate to prevent overfitting.

4.2 Training

The model is trained using the Adam optimizer with a learning rate of 0.0001 over 50 epochs, with a batch size of 32 and a 20% validation split. Sparse categorical cross-entropy is used as the loss function, compatible with the 0-based labels (0 to 29). An initial error, where the model expected 21 classes instead of 30, was resolved by updating the output layer to 30 classes. Hyperparameters, such as filter counts and kernel size, were tuned via grid search to optimize accuracy while maintaining low inference latency.

5. System Performance

5.1 Latency Tests

Inference on a single 128-timestep window takes less than 0.5 seconds on a 4-core CPU using TensorFlow 2.17.0. Data transmission involves 6 channels × 128 samples × 4 bytes, approximately 3 KB per authentication, fitting within the 20 KB/s bandwidth limit for a 1-second cycle. Total latency, including preprocessing and inference, meets the requirement of less than 1 second, ensuring employees are authenticated before reaching the turnstile.

5.2 Battery Usage

Smartphone sensor sampling at 50 Hz and Wi-Fi data transmission consume minimal power. The accelerometer and gyroscope use approximately 1 mW and 3 mW, respectively, resulting in less than 2% battery drain over an 8-hour shift for a typical 4000 mAh battery. This satisfies the requirement of less than 3% battery usage.

5.3 API Troubleshooting

If the API endpoint (http://localhost:5000/authenticate) returns a "Not Found" error, verify the following:
- The Flask server is running (execute `python door_controller.py`).
- The model file (`gait_cnn_model.h5`) is present in the `model/` directory.
- The POST request includes a valid JSON payload with a (128, 6) sensor data array.
- Port 5000 is free; if not, update the port in `door_controller.py` (e.g., to 5001).

6. Security Considerations

6.1 Spoofing

Gait patterns are inherently difficult to replicate without physical mimicry. The system's use of multi-sensor fusion, combining accelerometer and gyroscope data, increases the complexity of spoofing attempts, as attackers must match both signal types simultaneously. This enhances security compared to single-sensor systems.

6.2 Lost or Stolen Phones

The system requires a live sensor stream, preventing authentication with a stolen device unless the attacker replicates the owner's gait. To further mitigate risks, a device registration process with a PIN or biometric check could be implemented, ensuring only authorized devices initiate authentication.

7. Conclusion

The gait-based authentication system provides a reliable, low-latency solution for employee verification using smartphone inertial data. A 1D-CNN processes raw sensor data, achieving robust performance across phone orientations. The system meets stringent requirements for latency (<1 second), battery usage (<3% per 8 hours), and bandwidth (<20 KB/s). A corrected 30-class model ensures compatibility with the UCI HAR dataset, and API troubleshooting steps enhance operational reliability. Future enhancements, such as multi-sensor fusion or anomaly detection, could further improve security and adaptability.