Person Identification Based on Footstep Acoustic Signals

COMP 4531
IoT and Smart Sensing

Date
14-05-2025

Authors

Gonzalo Carretero	21158252	gch@connect.ust.hk
Inwook Kang	20898829	ikang@connect.ust.hk
Yoeng Kok Leong	21195066	klyoeng@connect.ust.hk
Elizabeth Sham	21198020	elsham@connect.ust.hk

I. Introduction

In today's world, there exists many ways to identify individuals, such as camera based identifications, biometrics and so on. However, most of the existing solutions to identifications require either expensive systems or direct user input. Behavioral recognition serves as a solution to forementioned problems. Behaviors such as walking, speech and gestures vary from a person to another. Footstep detection doesn't require any input from users and the required system can be built at a low cost. With recent developments on footstep identification, we saw a possibility of utilizing this for a real-time person identification system.

In this project, we utilize the intrinsic unique characteristics of a person's footstep's acoustic signals in order to identify different individuals in real time. Making use of acoustic signals serves as a less intrusive way of identification and is robust to visual occlusions or low light conditions compared to using cameras. The applications can span from integration into smart home/building security systems to improving robots awareness for personalized interactions as well as supplement other sensors for multiple object tracking under adverse visual scenarios.

Our system records the noise generated in an enclosed environment and sends it to our server for real time classification using a CNN model. We have collected our own data samples in order to perform identification of a reduced number of subjects.

II. Design

First iterations of ML Classifier design

We first tried to approach the problem of footstep detection and classification with a more hands on technique, writing algorithms to apply band pass filters into the audio signal and extract the peaks and times at which a footstep had occurred. We extracted 13 MFCC (Mel-Frequency Cepstral Coefficients) of the sliced footstep sounds computed and made use of a Nearest Neighbor Classifier (KNN) to find the closest neighbours of the coefficients of new footsteps recorded compared with the ones in the training samples. We found a K equal to 3 to perform strongly with a few recorded samples of different individuals walking.

The following picture shows our initial algorithm trying to detect the moments in time footsteps occurred within an audio:

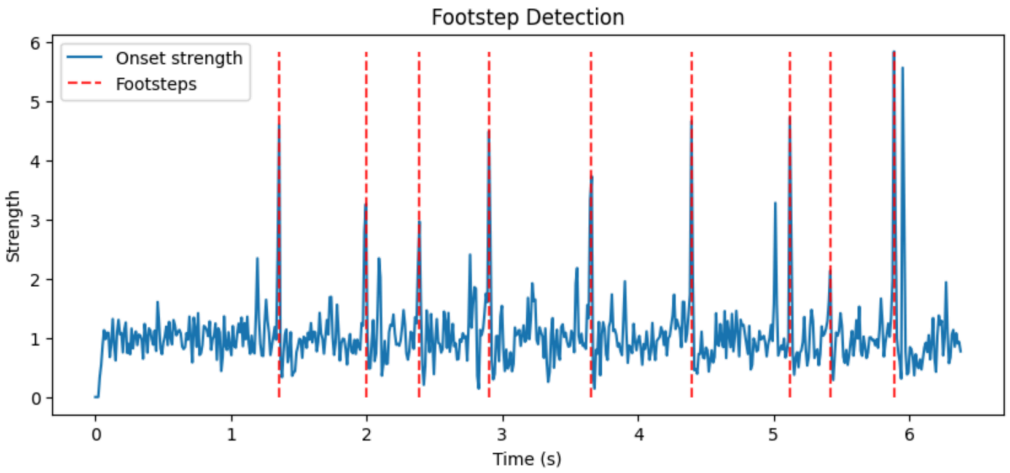

Figure 1: Footstep detection across a sample audio recording

In order to visualize the characteristics that are being extracted for each footstep by the MFCCs, and how the KNN is able to classify them, we performed a Principal Component Analysis (PCA) to reduce the dimensions of the coefficients to 2. In the following image, the orange dots represent the PCA-reduced features of one individual's footsteps, while the blue dots correspond to another person's footsteps.

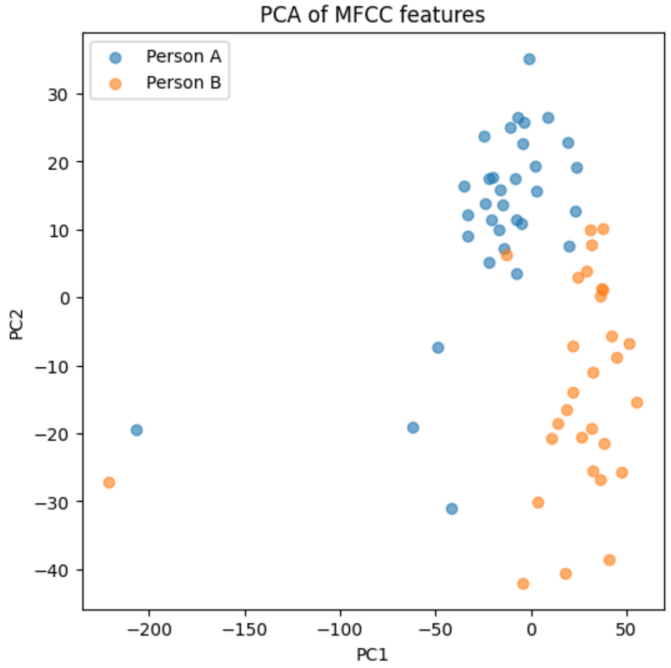

Figure 2: MFCC of the footsteps of two individuals after PCA

We can observe that the characteristics of the footsteps done by the two different people have a tendency to form a cluster and can be thus more accurately differentiated by our KNN model, obtaining very good results on new test audio data:

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9

    accuracy                           1.00        19
   macro avg       1.00      1.00      1.00        19
weighted avg       1.00      1.00      1.00        19

Confusion matrix:
[[10  0]
 [ 0  9]]
```

Figure 3: Test results of KNN on data of two separate people walking

We briefly explored combining knowledge of a person's footstep characteristics and the computed walking pace together with a simple Kalman Filter for more robust real time tracking of multiple people walking at the same time but later on discarded this approach as we didn't find significant results.

We decided to continue exploring more machine learning models to use for footstep identification. We conducted research on the performance of different types of models in footstep detection. We found that out of many ML models including Transformer, Recurrent Neural Network and Convolutional Neural Network (CNN), CNN performed the best. Therefore, we used a convolutional neural network for our final model design that is explained in more detail later on.

Our overall final system can be divided into the following modules:

A. Sensing Hardware

The following components are used for the collection of real-time raw audio data during deployment scenarios:

- Raspberry Pi
- Re-speaker

B. Signal Processing Pipeline

The raw audio data captured by the Respeaker microphones on the Raspberry Pi is streamed to a remote server through a socket using the TCP protocol.

The remote PC listens for incoming connection attempts. After successful connection with the Raspberry Pi, the server processes the incoming raw audio data by buffering and segmenting the data into .wav files of fixed 10 second duration.

These .wav slices are then fed into our machine learning model which performs additional data preprocessing and predicts the person who produced the recorded footsteps.

C. Training Dataset
Each of our team members' footstep audio files were used as the training set data.

We assigned the following labels for each member:

Gonzalo: 0, Tim: 1, Elizabeth: 2, Yoeng: 3

Training dataset has a total runtime of around 20 minutes; 5 minutes per person. We took our samples from a relatively controlled environment; floors were padded with mats, participants wore consistent footwears and outfit, and the environment had low background noise.

For recording, Gonzalo's Iphone XR was used with a sampling rate of 48000 Hz. Participants held the phone in their hand and walked with it at a usual pace and stride. Collected audio files were then labelled accordingly.

D. Data Preprocessing
Before feeding the data to the CNN model, the data is preprocessed in the following steps:

1. Raw audio files were loaded with the *librosa* library in python using a sampling rate of 48000 Hz. The raw audio files were then reorganized into frames, each frame consisting of 128 samples and overlapping 50% with adjacent frames.
2. Frames were further partitioned into segments of 128 frames to match the model input shape of (128, 128).
3. Each segment was then processed with the *librosa.feature.mfcc* function to obtain the mfcc spectrograms. The last segments were zero-padded to have consistent shapes.
4. Lastly, spectrograms were normalized.

E. Classification Model
We got inspiration on modern Face Recognition Systems in order to design a pipeline for model training and inference which is able to not only classify the people it has trained on, but that can also discriminate effectively when detecting a new type of user that is not in the system.

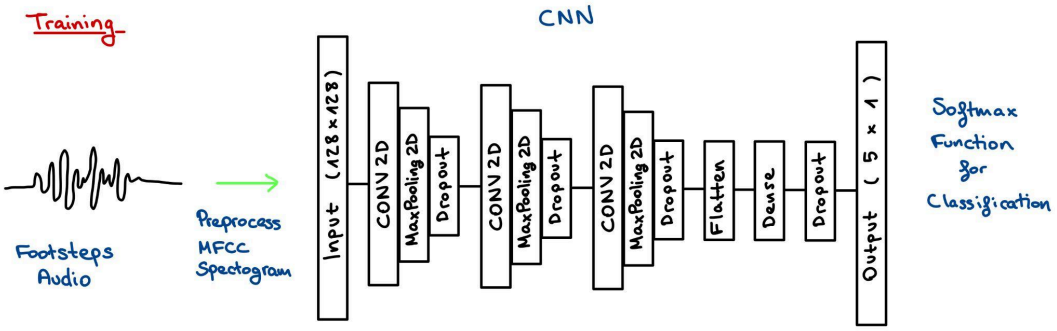

Figure 4: CNN architecture for training

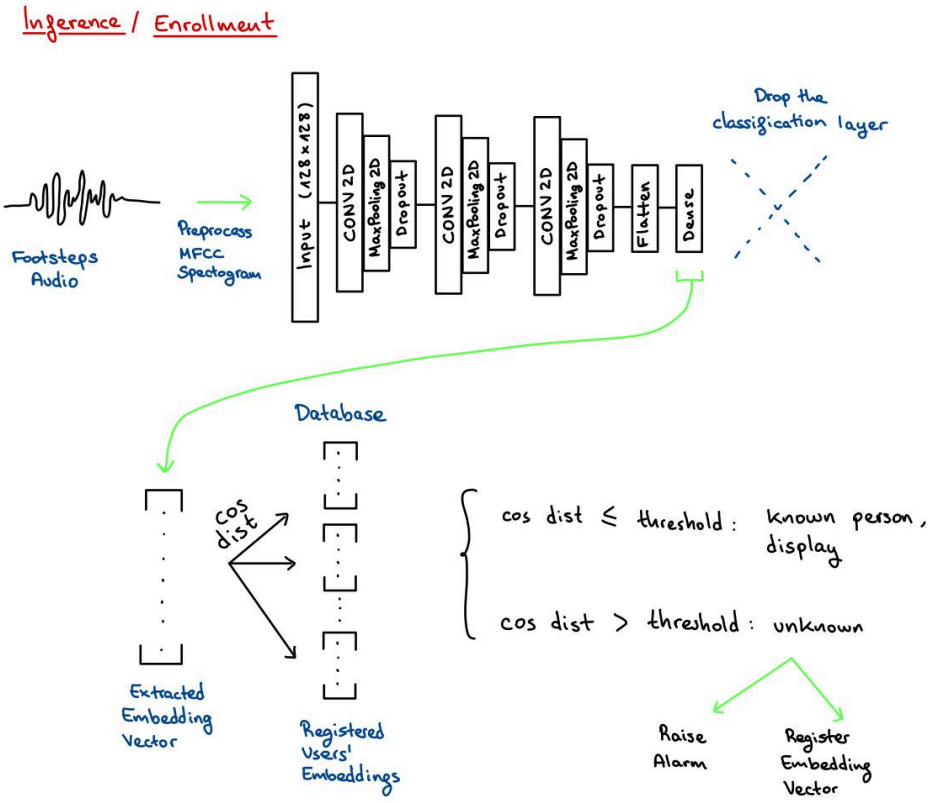

Figure 5: Model during inference or new person registration

After training a full CNN model to learn how to classify a set of people's footsteps with the samples we collected, we take all the embedding vectors of each person's samples (dense layer before the softmax) and save the **average** of them. This will serve as the unique id that represents a person for us.

Later during inference, we will pass one time the new processed audio through the model (without the classification layer), extract the embedding vector, and compare it using cosine similarity to the vectors we have saved for known people. If any of the similarity is greater than or equal to a threshold we have defined, then the system will identify the footstep as one of the registered classes, otherwise as "unknown".

We used tensorflow to design our model. The model layers are shown below:

```
model = Sequential()
# First Convolutional Block
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=input_shape))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))

# Second Convolutional Block
```

```
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))

# Third Convolutional Block
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))
# Fully Connected Layers
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

III. Implementation

Communication

The server and the Raspberry Pi share the same network. The Raspberry Pi sends an incoming socket connection request to the server. Once accepted, it collects the audio of its environment at a sampling rate of 48 kHz. It streams these audio bytes over to the server continuously in 1024 byte chunks. After running prediction on the audio files, the result is displayed on a tkinter GUI.

Model Training

With the prepared data and the model, we train the model for 30 epochs.
After training the models. We run the training data again through the model up until the dense layer with relu activation function (embedding model) to obtain the embeddings of each class. We then save the mean of embeddings for each class and these serve as id for each class.

Testing

The mean embeddings for each class are passed to and stored in the server. After the server connects to the raspberry pi, it's able to pick up sample .wav files. The audio files are preprocessed then passed to the embedding model to generate its embeddings vector. The samples' embeddings vectors are compared against the embeddings stored in the server, using cosine similarity, to either identify the person or classify the footstep as unknown.

IV. Challenges Faced

One of the challenges we faced was reducing long prediction latency during real-time identification to be suitable for real-world application. The latency bottleneck was located in

the model prediction process, which we tried to reduce by limiting the length of the audio the model was predicting on. This leads us to our next challenge: choosing a duration for audio input for optimal accuracy while maintaining fast inference times. Having more frames to predict from leads to better accuracy in exchange for longer prediction times. Our final challenge was determining a good threshold for confidence scores so unknown individuals do not get marked as known individuals. We initially relied on our softmax output layer for identification through the use of confidence threshold to detect unknown individuals but proved unreliable due to inconsistent confidence scores for unfamiliar footsteps. That is when we adopted the vector embedding extraction pipeline explained above, resulting in much better results.

V. Results

We first trained the model on 2 people: Tim (1) and Gonzalo (0), the result is shown in Figure 7. After trying the identification on 5 test footstep samples for each of us, we observed 100% accuracy in identification.

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.94	0.91	520
1	0.93	0.88	0.91	496
accuracy			0.91	1016
macro avg	0.91	0.91	0.91	1016
weighted avg	0.91	0.91	0.91	1016

Figure 6: training result of model trained on just 2 classes.

After seeing the successful result, we trained the model with all 4 classes. The performance is shown in Figure 7.

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.94	0.90	938
1	0.73	0.72	0.73	502
2	0.64	0.83	0.72	424
3	0.84	0.46	0.60	434
accuracy			0.78	2298
macro avg	0.77	0.74	0.74	2298
weighted avg	0.79	0.78	0.77	2298

Figure 7: training result of model trained on all 4 classes.

With good performance results for the model classifying all the people it was trained on, we proceeded to test it on identifying whether an unknown person is the one walking in the real time recording. We conducted several trials in order to fine tune the threshold of the cosine

similarity between the embeddings of the new footsteps and the registered user ones to decide whether it is close enough to any of them. We determined that a threshold of 0.9 for the cosine similarity resulted in correct predictions on both who is the known person walking as well as if it is an unknown person with an accuracy of around 90% including with some light background noise like people speaking.

VI. Discussion

The primary innovation of this project was the implementation of spectral data preprocessing which enabled the model to process smaller audio frames while retaining important frequency information. This contrasts with traditional approaches, which typically use larger frames or simpler feature extraction methods. Additionally, the use of sliding windows with overlap allowed us to achieve smoother transitions between frames, preventing spectral jumps that could negatively impact classification accuracy. These innovations collectively contributed to a more robust model, particularly for classifying softer sounds like footsteps. To build upon this experiment further, we can consider walking in different directions/motions and different footwear.

A limitation however, is that in environments with too much noise, the model performance significantly deteriorates due to insufficient data and lack of variations in data. If given more time to work on this project, we would train the model on more data and under more variations of conditions. However, we adapted to this situation by designing a model capable of rejecting strangers without training it on any "stranger" data samples for classification.

VII. Workload Contribution

Name	Contribution
Gonzalo	Identified and evaluated suitable machine learning models for footsteps detection. Researched and implemented ways to classify new footsteps as an unknown class Researched on MFCC theory and concepts of extracting distinct features to conduct model training. Co-implemented CNN-based model, including data preprocessing
Inwook	Reviewed relevant literature on model design and came up with the CNN model architecture. Researched and implemented ways to classify new footsteps as an unknown class. Co-implemented CNN-based model, including training routines. Fine-tuned learning model to improve accuracy through hyper parameters.
Elizabeth	Developed Python script to establish and manage TCP socket connection between the Raspberry Pi and remote PC. Ensure stable real-time audio streaming from device to server and processing to data files
Yoeng	Handled hardware component including the Raspberry Pi and ReSpeaker microphone Set up a Raspberry Pi environment and developed a Python script to continuously capture audio and transmit it over TCP connection. Implemented GUI to visualise real-time model inference results in user-friendly format.