Smartphone Sensing 2023
App 3, Option 2
group 9
Samsung Galaxy A13, Android 13

1 Load balance

1.1 Tony Yang, 5651794

- Android Method
  - Online RSS-KNN location classification
  - 1D Spectral Contrast
- Off-line Method
  - Explore novel ML techniques
- Tasks and Deliverable
  - Gather RSS & acoustic data

1.2 Giacomo Mazzola, 4872673

- Android Method
  - Fix and Improve CNN model
- Off-line Method
  - Fix and Improve CNN model
- Tasks and Deliverable
  - Gather acoustic data

2 Introduction

We present "AERSS", our final app, designed to accurately detect locations across 16 cells on the 2nd floor of Building 28. This is achieved by utilizing both `RSS` and `Acoustic Echo Recognition`. **Please note that the gained accuracy will all be summarized in Section 5.**

3 Acoustic Signals & Feature Extraction

3.1 Description

Our probe signal design and echo recording method remain the same as described in the second report. For feature extraction, a `spectral contrast` 1D array is computed, mirroring the magnitude spectrogram's length. Each contrast array position is given a value—the difference between the maximum and minimum values in the same frequency bin of the magnitude spectrogram. This method is favored for its robustness against noise, a necessity due to the substantial random noise generated by the phone's low-quality embedded microphone.

The outcome of feature extraction is `immediately displayed on the GUI`. As demonstrated in figure 1,

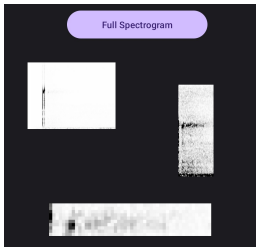

Figure 1: Part of GUI which shows three spectrograms

the three spectrograms correspond to the complete 500ms recording, the extracted 150ms recording (after omitting the chirp signal), and the further extracted spectrogram from 12kHz to 13kHz.

It's worth noting that the `spectrogram contrasts are plotted relatively` to highlight the patterns. As illustrated by the top-left spectrogram in figure 1, the magnitude of the echo is minuscule compared to the chirp signal. Therefore, we adjust the contrast for each spectrogram relatively. This is achieved by identifying the maximum and minimum values in the spectrogram and then reassigning magnitude values.

3.2 Unsuccessful attempt

We considered and attempted to use `MFCC`, but given its complex processing requirements on the magnitude spectrogram, we could not implement it successfully.

Each value in the magnitude spectrogram was initially multiplied by a `"magic factor"` of 2.4 and capped at 255 with the intention of `further enhancing the pattern`, making it easier for our deep learning models to identify the pattern. However, we later discovered that this amplification also intensified the noise, causing a slight dip in accuracy. We then considered training the model with the enhanced spectrogram and testing it with lightened ones (adjusting the magic factor to around 1.5 and capping each value at 255). This, however, also negatively impacted accuracy. Consequently, we abandoned the idea of further enhancing the spectrogram, keeping it relatively untouched after plotting.

3.3 Novelty

- 1D Spectral Contrast
- Real-time plotted spectrograms
- Relative contrast spectrograms

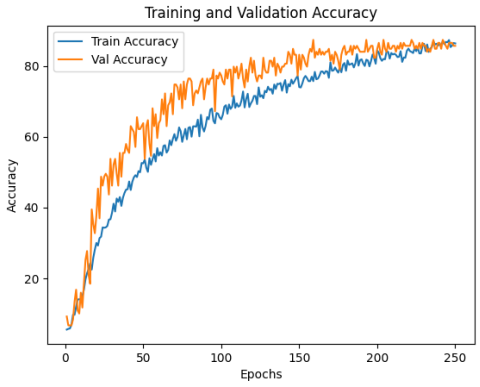

Figure 2: Comparison between testing and validation accuracy for 16-cells CNN model

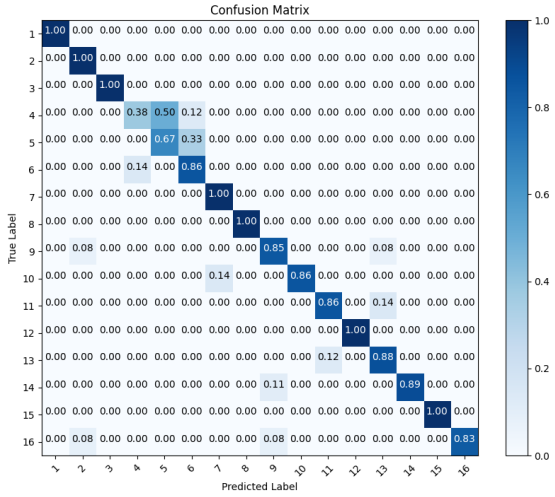

Figure 3: Confusion matrix for the 16-cells CNN model

4 Machine Learning Models

4.1 Description

After gathering acoustic data for all cells, the model was extended to classify all the 16 cells. The model was experiencing over-fitting at first, which was mitigated quite well by introducing a `drop_rate=0.2` in between the two fully connected layers. Figure 2 reports the testing and validation accuracy to prove that over-fitting is minimal, if not absent, in the new model.

The accuracy achieved by validating the model offline is 86%. By inspecting the confusion matrix in figure 3, it can be noticed that the major reason why the accuracy is not close to 100% is explained by the low accuracy between classes C4, C5 and C6, which are identical spots on an internal staircase of building 28. Our novelty approach fixed this issue as described in the following section.

4.2 Unsuccessful attempt

The main problem of this model is that, when exported to android, it features an accuracy close to random. To solve this problem, we tried multiple approaches of which:

- Migrate the model to TensorFlow
- Use the ONNXRuntime [1] library to export the CNN to android without compressing the model, as opposed to the PyTorch JIT method, which results in changes to the network.

Unfortunately neither of these approaches worked as intended. Therefore, we fixed this problem by introducing the novel approach described in the next subsection.

We also attempted to implement `BNN` using the `Larq` library in Tensorflow. However, we encountered issues with weight binarization and therefore had to abandon this approach. We also tried collecting data at a very high speed, as suggested in Qun's paper, but it proved ineffective. To ensure that each data point gathered contains 150ms of usable audio information, we adjusted our collection pace to approximately one datum every 2 seconds.

4.3 Novelty

Instead of deploying one CNN that classifies 16 cells with lower frequency, we deploy two CNNs that classify less classes with higher accuracy. We also deploy two KNN models trained on RSS data. The first KNN model is used to detect the side of the building the user is located, either west or east, and based on this, one of the two CNNs is selected to be fed with the spectogram. The second KNN is used to achieve high accuracy for cells C4, C5 and C6. when one of these cells is classified by the CNN, since they are characterized by very similar features, we use the KNN to determine the floor the user is located at and therefore, determine the accurate cell.

This approach was chosen because we found that a CNN trained with data from 9 cells performed excellently. However, when data from a 10th cell was added, it compromised the model's performance. Therefore, we opted for deploying two separate CNN models. This strategy also leverages the "KNN+RSS" knowledge from class, aiding in identifying the building area and deciding which model to activate.

For the purpose of gathering acoustic data, a spinner was designed, including the names "C1" to "C16", as depicted in figure 4. Moreover, a separate app, shown in figure 5, was created for the purpose of gathering RSS data and arranging them in the format we desire. Detailed source code can be found at [2].

We've designed a user-friendly Python script for training purposes [1]. It's quite simple that all data is placed into a directory, with each subdirectory representing a unique class. After defining the proportions of validation and testing data, the training process initiates automatically and independently saves the model. Given the relatively small size of the collected data, we allocated 80% of it for training, with the remaining 20% split evenly between validation and testing.

[1] https://onnxruntime.ai/docs/get-started/with-java.html

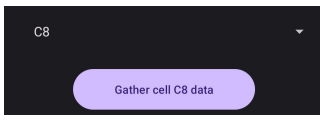

Figure 4: Part of GUI dedicated to collecting data

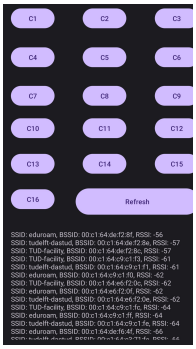

Figure 5: Independent app dedicated to gather RSS data

5 Indoor Localization

5.1 Description

As discussed in the previous section, we have deployed four ML models online. To allow users to test all our models, we created two switches. Users can choose to "only use CNN trained over 16 cells," "activate RSS on top of CNN trained over 16 cells," or "activate RSS + 2 CNN models," as illustrated in figure 6.

We also simulated app evaluation. We tapped the tracking button five times and recorded the accuracy of each cell using all three options, as displayed in figure 7. The "Duo CNN + RSS" configuration evidently achieves the highest average accuracy. However, it has the drawback of excluding cell C10 because its inclusion disrupted the overall model during training. Further adjustments to rectify this issue will be reported during the evaluation phase.

Please note that all the features and methods discussed so far, including the feature extraction methods and the ML models (2 CNN + 2 KNN), are `implemented online`. However, it's worth mentioning that our model currently lacks robustness against interference. Nonetheless, as we will discuss in the following section detailing our work plan for the week prior to the evaluation, it's highly probable that our model will be able to tolerate a certain level of interference.

5.2 Unsuccessful attempt

In using KNN + RSS, our initial idea was to represent the west and east sides of the building as two distinct points. After each Wi-Fi scan, the area with the smaller Euclidean distance was designated as the current location. However, this method proved ineffective. A thorough investigation wasn't carried out, but our intuition suggests that the Euclidean distance measure is more effective when points have overlapping Wi-Fi access points. Calculating the Euclidean distance between two MAC addresses does not yield an accurate "real distance". We resolved this issue by using KNN

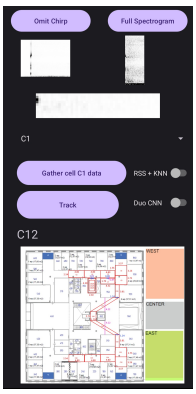

Figure 6: Final App

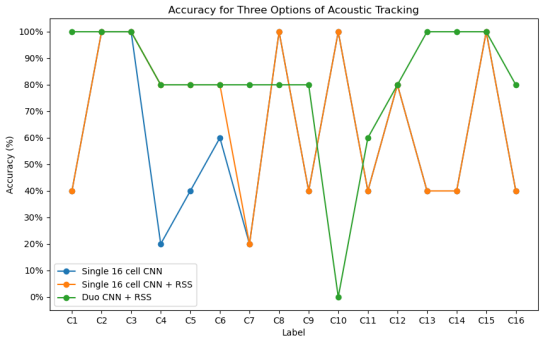

Figure 7: Accuracy comparison

to identify 16 cells. If the result lies between 1 to 10, it is classified as west, and east otherwise.

5.3 Novelty

- Flexible options for users

6 Future Work

In the following week before the evaluation, we are going to try to improve even more the app by working on the following points:

- Collect more samples. In fact, because we changed the way to gather samples, we had to start collecting them from scratch. Our goal is to reach at least 1000 samples for each cell. Including samples with more disturbance from the environment.
- Try to fix the current problems with the CNN and make it reach a high accuracy classification on 16 cells.

References

[1] Tony Yang & Giacomo. *Phone-Sensing-ML*. `https://github.com/tonyyunyang/Phone-Sensing-ML`. Accessed: 2023-06-12. 2023.

[2] Tony Yang. *Android-Gather-WIFI-New*. `https://github.com/tonyyunyang/Android-Gather-WIFI-New`. Accessed: 2023-06-12. 2023.