Exploring Audio Sensing in Detecting Social Interactions Using Smartphone Devices

Jon Baker

This thesis is submitted on December, 2019 for the degree of Doctor of Philosophy in Electronic Engineering

Abstract

In recent years, the fast proliferation of smartphones devices has provided powerful and portable methodologies for integrating sensing systems which can run continuously and provide feedback in real-time. The mobile crowd-sensing of human behaviour is an emerging computing paradigm that offers a challenge of sensing everyday social interactions performed by people who carry smartphone devices upon themselves. Typical smartphone sensors and the mobile crowd-sensing paradigm compose a process where the sensors present, such as the microphone, are used to infer social relationships between people in diverse social settings, where environmental factors can be dynamic and the infrastructure of buildings can vary.

The typical approaches in detecting social interactions between people consider the use of co-location as a proxy for real-world interactions. Such approaches can under-perform in challenging situations where multiple social interactions can occur within close proximity to each other, for example when people are in a queue at the supermarket but not a part of the same social interaction. Other approaches involve a limitation where all participants of a social interaction must carry a smartphone device with themselves at all times and each smartphone must have the sensing app installed. The problem here is the feasibility of the sensing system, which relies heavily on each participant's smartphone acting as nodes within a social graph, connected together with weighted edges of proximity between the devices; when users uninstall the app or disable background sensing, the system is unable to accurately determine the correct number of participants.

In this thesis, we present two novel approaches to detecting co-located social interactions using smartphones. The first relies on the use of WiFi signals and audio signals

to distinguish social groups interacting within a few meters from each other with 88% precision. We orchestrated preliminary experiments using WiFi as a proxy for co-location between people who are socially interacting. Initial results showed that in more challenging scenarios, WiFi is not accurate enough to determine if people are socially interacting within the same social group. We then made use of audio as a second modality to capture the sound patterns of conversations to identify and segment social groups within close proximity to each other. Through a range of real-world experiments (social interactions in meeting scenarios, coffee shop scenarios, conference scenarios), we demonstrate a technique that utilises WiFi fingerprinting, along with *sound fingerprinting* to identify these social groups. We built a system which performs well, and then optimized the power consumption and improved the performance to 88% precision in the most challenging scenarios using duty cycling and data averaging techniques.

The second approach explores the feasibility of detecting social interactions without the need of all social contacts to carry a social sensing device. This work explores the use of supervised and unsupervised Deep Learning techniques before concluding on the use of an Autoencoder model to perform a Speaker Identification task. We demonstrate how machine learning can be used with the audio data collected from a singular device as a speaker identification framework. Speech from people is used as the input to our Autoencoder model and then classified against a list of "social contacts" to determine if the user has spoken a person before or not. By doing this, the system can count the number of social contacts belonging to the user, and develop a database of common social contacts. Through the use 100 randomly-generated social conversations and the use of state-of-the-art Deep Learning techniques, we demonstrate how this system can accurately distinguish new and existing speakers from a data set of voices, to count the number of daily social interactions a user encounters with a precision of 75%. We then optimize the model using Hyperparameter Optimization to ensure that the model is most optimal for the task. Unlike most systems in the literature, this approach would work without the need to modify the existing infrastructure of a building, and without all participants needing to install the same app

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or am concurrently submitting, for a degree or diploma or other qualification at the University of Kent or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or is being concurrently submitted, for any such degree, diploma or other qualification at the University of Kent or any other University or similar institution except as declared in the Preface and specified in the text. This thesis does not exceed the prescribed limit of 60 000 words.

Jon Baker

December, 2019

Acknowledgements

I would like to thank my supervisor Christos Efstratiou for his continuous support and guidance throughout my PhD. This PhD was probably the most ambitious task I have undertaken, and there is no way I would have been able to do it without your direct guidance. I learned so much, and I will be forever grateful for all of the skills that I have acquired.

A special thank you to Sze Pang, who had to deal with all of my stress. Thank you so much for supporting me in everything that I do, for cooking me dinner, washing my clothes and for convincing me to carry on when I had enough. Without you by my side, I would have never finished this thesis.

Thank you James Alexander Lee for helping me through all of the negatives. For supporting me whenever my accuracy was terrible and for teaching me to be more positive about everything.

Thanks to Deogratias Mzurikwao, Hamed Alsufyani and Leo Valberg for all of their help and discussions about Machine Learning.

Thanks to the University of Kent for awarding me their 50th Anniversary Scholarship, which helped to fund the first three years of this PhD.

And thank you to the team at Researcher, the company who employed me during my write-up year. Thanks for getting behind me, giving me extra space to undertake the rest of this PhD and for helping yet another scientist make their breakthroughs.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In a technological world where smartphones devices and wireless connectivity are becoming more common, there is a significant scope to develop methodologies for the gathering of hard data about the behavioural activities of users within diverse social environments, for the purpose of life-logging, memory augmentation and social sensing. It's possible to imagine an automated system which can track the ongoing social interactions between a set of users in urban environments whilst branching away from energy-consuming Global Positioning System (GPS) [10] methodologies and instead use new techniques which can passively track with less power consumption, without requiring visibility of the sky, and continuously.

There is an increasing interest in using technologies that can capture, record and analyse the daily activities of people [11]. From the perspective of self-improvement, the quantified-self movement[1] is aiming to offer ways for individuals to record and understand their own behaviour. In a societal scale, tracking the activities of people can allow the analysis of the behaviour of whole communities, and enable large-scale analytics [12]. These visions are primarily motivated by the proliferation of life-logging technologies that can capture events, experiences, and raw data, whilst keeping them in a chronological time-line [13]. The "life data" that is involved in life logging, can be acquired using devices such as sensors or actuators, where it is then

[1]Quantified self is a movement that incorporates technology to acquire data by self-sensing various aspects of an individual's life, with an aim to improve self-awareness and human performance.

stored and examined. Recent technologies can be used to develop self-monitoring and self-sensing devices for life logging, which would deliver information of behavioural changes, quantified-self and more [14].

Social sensing describes the capture and utilisation of social contexts during everyday interactions. It allows the sensing of social situations dynamically, such as conversations, team activities, meetings and so on [15–17]. More recently, social sensing has allowed the detection of more sophisticated interactions, such, conversational flows, co-location patterns and emotion. Sensing specifics requires more complicated logic to deliver accuracy, which unfortunately come at the expense of power. Due to the portable nature of a smartphone, there is an opportunity to use this device as a wireless life-logging device through the creation of an app, which can make use of a background service that can track users as well as the implementation of a graphical user interface that can visualise the life-data recorded by the user tracking. However, life-logging devices are typically battery powered and would require continuous sensing, meaning that power consumption would be an important issue to tackle. Therefore, the app would need to contain techniques that are accurate and remain stable across the duration of a user's daily life.

Continuous sensing systems can be used to increase our understanding of human behaviour, health and much more. This thesis investigates how that might be possible by utilising the sensors on-board of a smartphone device to capture the social co-presence of humans in real time using smartphone devices. There are two main aspects: (1) the accurate tracking of social interactions in different scenarios and (2) the scalability of the tracking system, which can be utilised to capture the social context from other participants who are not using the sensing system.

Context aware computing is an approach in Pervasive and Ubiquitous Computing, where technology can be aware of the "context" of the user and adapt to that context [18]. It can enrich the capabilities of intelligent and smart devices. It works by collecting situational information about the environment and is used to enrich user-experience based on the information collected. For example, current smartphone devices will have screens which can auto-adjust the brightness in accordance to

the light levels around the user and can even rotate the user-interface automatically based on an angle in which the user is holding the device. When collecting information, devices need to be wireless, powered by battery and contain relevant sensing technologies. A smartphone device is a suitable match for this criteria and can be used for building context aware systems. It is presumed that most people will always carry their smartphone device with them wherever they go, and these devices typically contain a variety of sensing technologies, such as accelerometer, GPS, gyroscope, magnetometer, microphone and proximity sensor. These technologies can each contribute to the recording and collecting of situational information about the environment. This means that a large portion of smartphone devices have access to the internet, without restrictions, which would ultimately allow situational information to be easily transferred or shared to a central data storage point, and thus makes smartphones an even greater context aware system. The question here is whether any methods of human co-presence detection could be utilised into a real-world environment, on a large scale deployment, without any constraints and minimum intrusiveness.

Android is a mobile operating system developed by Google. It comes installed on a variety of smartphones and tablets, and is the most used mobile operating system [19]. The benefit with Android is that the smartphone device will typically support a number of wireless communication methods which are able to wirelessly collect and transmit data. Android devices are rechargeable and can compute in the background whilst the device is not being used by the user. iOS is another popular mobile operating system solely for smartphone manufacturer "Apple". Unfortunately, background processing in iOS is highly regulated, with only certain types of background processing allowed. Android and iOS are essentially two different platforms, and while Android is fairly straightforward to deal with, iOS is not. Building a sensing app which works for both platforms is problematic, and therefore the majority of work carried out in this thesis makes use of only Android smartphone devices.

For social sensing, the sensor technologies inside smartphone devices can be used to employ device discovery, which can be used to interpret as human co-presence. Further to this, the device can also sense the environment around itself; not only can

the device detect if two users are interacting, but if they are indoors, outdoors, in the rain, in a humid environment or even participating in certain activities. Whilst outside, devices can capture the GPS location available from active satellites to determine the positioning of the device within the world. However, in indoor urban spaces, with no access to the sky, GPS stops working and typical approaches will instead use the signals transmitted from nearby WiFi access points to measure how far a device is from a the access point; if two devices are the same distance from the access point, it can be inferred that they are co-located.

Audio is another modality of smartphone sensing, since microphones are typically within the device and can record in the background to periodically collect audio samples. The great thing here, is that the microphone can be accessed at any time, as long as the user has accepted a permission on the smartphone. And the device can record as much as it likes, storing the raw audio data in lossy or lossless methods, or extracting auditory features on the fly. Moreover, the continuous sensing of audio can be facilitated by typical duty cycling techniques [20], which can be applied to reduce the battery consumption whilst maintaining the accuracy of a continuous sensing system.

A recent addition is the mobile system-on-chip technology [2] which increases the performance of Android devices. Machine learning solutions are highly useful when packaged into Android apps and running them on the mobile platform, however, this technique has computational overhead on device CPU and can drain battery. To overcome these issues, there are a number of attempts to use hardware acceleration by using GPUs or DSPs [21, 22], with good energy consumption. Huawei, Samsung, MediaTek and others have developed dedicated AI chipsets to help make AI affordable, effective, and reliable whilst sufficiently powerful computing power. These chipsets claim to accelerate computer vision, natural language processing, and additional machine learning tasks in smartphones. There are even apps dedicated to benchmarking the performance of an AI chipset within a device [23].

[2]System-on-chip is the term used to describe a complete processing system contained in a single package. It combines multiple components into a single chip on a smartphone device to save space, cost, and power consumption. This single chip also connects to other components, such as cameras, display, RAM, and flash storage

Machine learning can improve the performance of real-world applications [24] by using a mathematical models to train data to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a vast selection of tasks including computer vision and audio processing.

Deep Learning is a subfield of machine learning, and involves the usage of algorithms inspired by the structure and function of the brain. More specifically, these are known as neural networks. In deep learning, a computer model is trained to perform classification tasks directly from input data such as text, images, sound or signals. Models are typically trained using a large data sets compared with typical machine learning, which consists of labelled data multi-layer neural network architectures.

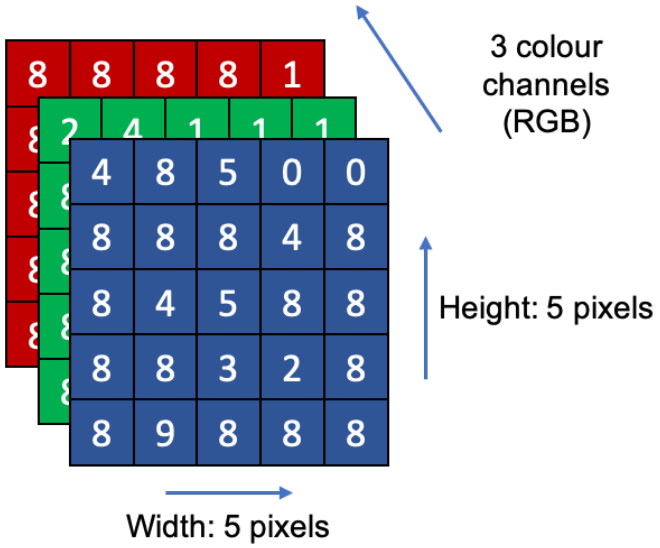


Figure 1.1: Overview of an bitmap image input for a neural network, represented by 3 channels of Red, Green and Blue with a width and height of 5 and each value representing a pixel in the bitmap

Convolutional neural networks (CNN) are a deep learning algorithm that can take an input image and assign weights/bias (importance) to various sections of the image. As humans, we are are able to look at a photograph of a street, and we are able to identify which parts of the street are road and which parts are cars. A CNN can be useful in teaching a machine to identify the separate components, using digital image data. For example, if we imagine that we have a 2D image, such as a bitmap (see: figure 1.1), this image can be represented as a vector which has a width, a height and a depth (for the image's RGB values). This data can be used as the input for

a CNN, which is trained to learn a specific classification task, such as identifying whether the image is of a car or not.

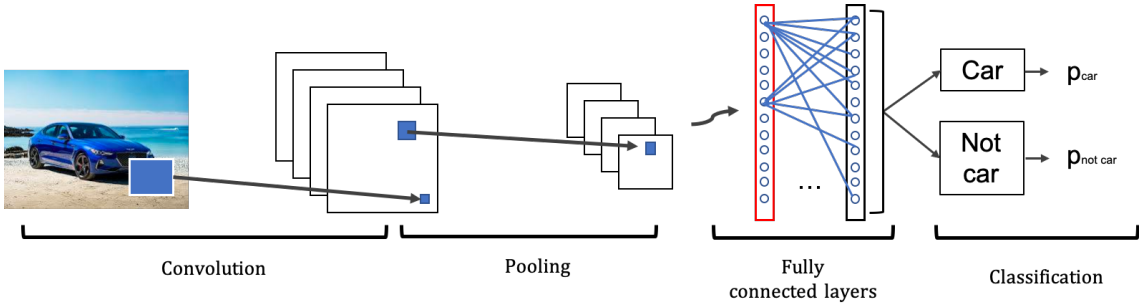


Figure 1.2: Overview of an example Convolutional neural network, showing the 4 stages: convolution, pooling, fully connected and classification. In this example, the model is trained to identify a binary decision: whether or not the input image is a car

A CNN typically starts with a convolution stage (as described visually in figure 1.2), where we can extract a small patch from this image and then run it through a tiny neural network to produce N outputs (known as filters or kernels). If we slide that little neural network across the image without changing the weights, we end up with another drawn image with a different width, a different height, and more importantly - a different depth. Instead of just R, G and B, now we have an output that's has K colour channels. After this convolution stage, the next is pooling. Pooling is a technique used to reduce the spatial size of the convolved feature; to decrease the computational power required to process the data through dimensionality reductions. The next step is to feed the output of the pooling layers into Fully-Connected layers. Using Fully-Connected layers is a generally a way for the model to learn a non-linear function in that space. And finally, the output is flattened and classified using the Softmax Classification [25] technique, which in short turns the numeric output of the Fully-Connected layers into probabilities of classes. Over multiple epochs, the model should be able to distinguish between dominating and low-level features and use this to classify the contents of an image.

With speech processing, the same thing can be applied and we can use raw speech for a 2D CNN architecture with little pre-processing. In this methodology, instead of having a depth of 3 like a typical bitmap, we use a depth of 1 which contains a single-channel image consisting of audio data. The data for this instance would be

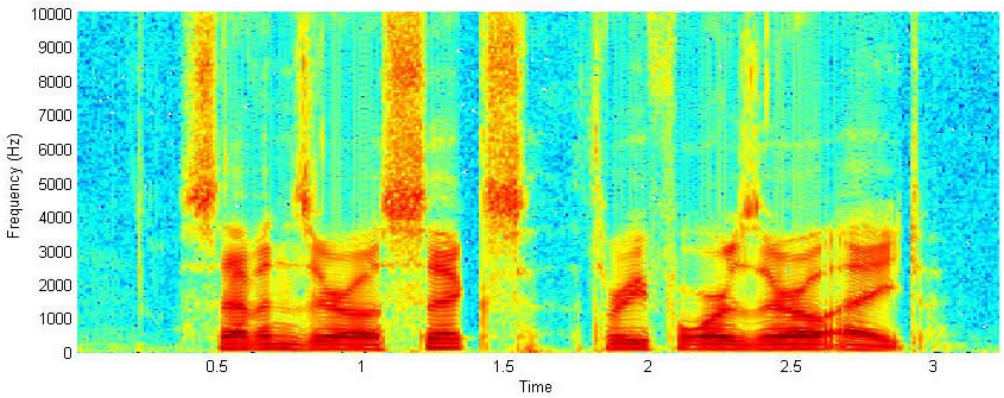


Figure 1.3: Example of a 2D Spectrogram from a sample of audio. In this figure, the x-axis is the duration of the audio track in seconds and the y-axis is the frequency of the audio signal in Hertz

the 2D Spectrogram data of the audio frequencies, where the X-Axis represents time and the Y-Axis represents frequency in Hz. An example can be found in figure 1.3.

A CNN is not the only method available. Alternatively, there are also Graph Neural Networks (GNN) — a neural network that can learn from graph-structured data, which has no notion of rotation or translation of it's input. GNNs are typically used to classify much larger images [26].

1.1 Thesis Contributions

This dissertation explores the sensing of human social interactions using smartphone devices. The principal contributions of this thesis are:

- A collaborative sensing system designed for Android devices entitled 'Next2me', which aims to capture the social interactions performed by social groups within close proximity to each other, but separate by nature. The system is deployed as a smartphone app that users can install onto their Android devices, and works by automatically collecting data about the signal strengths of nearby WiFi access points to determine if people are co-located. Once the system determines that people are co-located, a cloud service uses the similarity of *audio fingerprints* captured by the smartphone microphones to separate social groups. This system does not require the training on voice samples of participating users and yields high precision in these dense social scenarios.

- A separate and scalable continuous tracking system entitled 'Speaking2Me', which aims to capture social information without the need for all social contacts to use the technology. This system captures social information by using the audio signals only, which can operate on a single smartphone device only. The system makes use of a deep learning autoencoder to extract encoded representations of people's voices and then calculates speaker identification and speaker counting using a trained binary classification neural network model. This system is designed to minimise memory footprint of the trained model, and does not require the training of voice samples from participants social contacts.

These systems have no reliance on the existing infrastructure of buildings in which conversations may occur, and can work anywhere. There is also no need to train these systems with new data from the social contacts of participants. These are both social sensing solutions which are scalable and are easy to deploy.

1.2 Thesis Structure

This thesis is structured into the following chapters:

Chapter 2 presents a literature review exploring the various aspects of sensing using smartphone devices. From WiFi sensing, audio sensing, to multiple modal methods of sensing. The challenges of these methodologies then motivate the review of the technologies used to passively detect social interactions, including the use of Machine Learning.

Chapter 3 presents a study exploring the use of a continuous sensing system to passively track and collect data about the social interactions which occur during everyday life. This section of the thesis dives into a new study that was conducted to test the implementation of a social sensing mobile app system based on WiFi and Audio Signals called "Next2Me", where the participants ran the app on their Android smartphone devices. An analysis identifies the key social groups which occurred throughout experiments, and provides a precision performance of 88% within noisy environments, including any smartphones that are placed in users' pockets - whilst

maintaining a very low energy footprint (less than 3% of battery capacity per day).

Chapter 4 looks into the scalability issues of the Sensing System described in Chapter 3, where all participants are required to use the same technology. This chapter also describes the construction of a new sensing system which uses a Speaker Identification methodology to count the number of participants from within a social interaction using Deep Learning and Smartphone devices. This method makes use of an autoencoder-based deep learning approach to produce an technique which can operate in an unsupervised manner without the need to use training data from the speech of social contacts from participants. This is followed by a binary logistic classification to improve accuracy when compared to a threshold based classification. In 100 generated social scenarios, the Speaker Identification system performs at 74% precision.

Chapter 5 presents a summary of the work conducted in this thesis and discusses the contributions made to research area and concludes on the research questions and contributions made in this thesis.

1.3 List of Publications

[27] Jon Baker and Christos Efstratiou. Speaking2Me: Detecting Social Interactions using Speech Signals and Deep Learning in Smartphone Devices, 2019, MobiUk 2019.

Jon Baker and Christos Efstratiou. Capturing social interactions through smartphone devices using WiFi and audio signals. Poster presented at the EDA School Research Conference, 2018.

[28] Jon Baker and Christos Efstratiou. Next2me: Capturing social interactions through smartphone devices using WiFi and audio signals. In Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pages 412 - 421. ACM, 2017.

[29] Chiara Lunerti, Richard M Guest, Ramon Blanco-Gonzalo, Raul Sanchez-Reillo, and Jon Baker. Environmental effects on face recognition in smartphones. In 2017 International Carnahan Conference on Security Technology (ICCST), pages 1 - 6.

IEEE, 2017.

[30] Chiara Lunerti, Richard Guest, Jon Baker, Pablo Fernandez-Lopez, and Raul Sanchez-Reillo. Sensing movement on smartphone devices to assess user interaction for face verification. In 2018 International Carnahan Conference on Security Technology (ICCST), pages 1 - 5. IEEE, 2018

CHAPTER 2

Literature Review

This literature review focuses on the technologies used in Social Sensing using smartphone devices. It covers the many technologies used in location and co-location techniques by single modality or multiple modality approaches.

For single modality, this review looks into the use of Global Positioning Systems (GPS) for detecting when people are close to each other. It also reviews Signal Strength approaches using trilateration or triangulation, Location Fingerprinting techniques using Bluetooth or WiFi, and the use of WiFi Probes for infrastructure-based techniques.

For Multiple modalities, this review looks into the use of Bluetooth combined with WiFi, the use of accelerometer to enrich data sets, and the use of wearable-technologies to capture social data about the wearer.

Finally, this literature review covers the use of Speech Processing, to explore how Voice Activity Detection is used to detect when speech occurs within an audio stream, and then how Speaker Identification is utilised to identify a speaker from data, which represents an individual's speech biometrics.

Approach	Modalities	Target	User Interface	Performance
RADAR [31]	WiFi	Windows 95 laptops	N/A	3.5m error distance
WiFiScan [32]	WiFi	Android app	Indoor map	2m error distance
SpyLoc [33]	Microphone WiFi	Smartphone	None	1m error distance
Vanderhulst et al. [34]	WiFi	WiFi infrastructure	Smartphone app	0.96 F-Score accuracy
Wang et al. [35]	WiFi	WiFi infrastructure	None	93.9% detection accuracy
Comm2Sense [36]	WiFi	WiFi RSS analysis	N/A	50th percentile error: 0.5m
Virtual Compass [37]	Wi-Fi Bluetooth	Smartphone	Windows Mobile App	50th percentile error: 0.9m, 90th percentile: 2.7m

Table 2.1: Comparison of reviewed proximity/distance estimation-based systems

2.1 Single Modality

Smartphones with an embedded GPS sensor are being increasingly used for location determination, which enables location-based services that can deliver location context to map apps. When using location data, it's possible to compare the co-ordinates of two devices to deem if they are both co-located or not. The GPS sensor on the smartphone devices provides adequate accuracy, however, the main limitations are that it costs such a high energy consumption and is unavailable in locations where the sky is obscured and view of GPS satellites is not available [31]. The transmitted GPS signals also exhibit strong linearity, and are thereby subject to diffraction and reflection by buildings [2].

This section reviews alternative modalities to the use of GPS for location-based systems. WiFi is explored due to it's low-powered and continuous sensing possibilities. Many WiFi systems exist, including various ways of analysing the signal strength received from nearby WiFi access points. These are compared in table 2.1. This section also looks into the use of Ultrasound and how this methodology can be used for localisation within buildings, and we take a look at Bluetooth technologies

embedded in off-the-shelf smartphones and how they are used for indoor-localisation techniques.

2.1.1 Comparison Metrics

When comparing data collected from multiple devices and inferring distance between each device, it's not exactly plausible to infer this 'distance' as meters or centimetres. Therefore, a common practise is to use a distance or similarity metric, such as Euclidean distance or Manhatthan distance.

Let's assume that we have two data sets of WiFi access point signal strengths collected from two smartphone devices and received during a 5-minute window. The Euclidean distance between two vectors p and q is defined as:

$$d\left(p,q\right)=\sqrt{\sum_{i=1}^{n}\left(q_i-p_i\right)^2}$$

where d is the distance of the straight line between two points (p,q) in Euclidean space. With this metric, we can assume that a distance of 0 means both devices were extremely close between this 5-minute capture window, because the vectors are identical. Whereas, the larger the distance, the further away the vectors are in the Euclidean space, and thus the devices are likely further away from each other, because the vectors are more dissimilar.

Manhattan distance on the other hand, can be defined as the distance between two points measured along axes at right angles:

$$d(p,q)=\sum|q_i-p_i|$$

Manhattan Distance is occasionally preferred over the use of Euclidean distance in machine learning tasks when the dimensionality of the data increases.

Alternatively, for the same task, you might want to use a similarity metric to compare the vectors. The Jaccard coefficient is a measure of similarity between two sets. For

two sets p and q the Jaccard coefficient can be defined as :

$$J(p,q) = \frac{|p \cap q|}{|p \cup q|}$$

The Jaccard coefficient will be equal to 1 if the sets are exactly the same, and 0 if they are very dissimilar.

2.1.2 Signal Strength Trilateration Approaches

Trilateration is a technique that uses the known distance from at least three fixed points in 2D space to calculate the position of a user. In this case, the fixed points are represented as signal emitting WiFi Access Points or cellular towers, as outlined in figure 2.1.

In [32], the authors design a system for calculating the exact location of user given the exact location of WiFi access points and distances from each access points to user, which relies on a minimum of 3 access points to work. They note that their system performs with 2 meters accuracy.

2.1.3 Signal Strength Triangulation Approaches

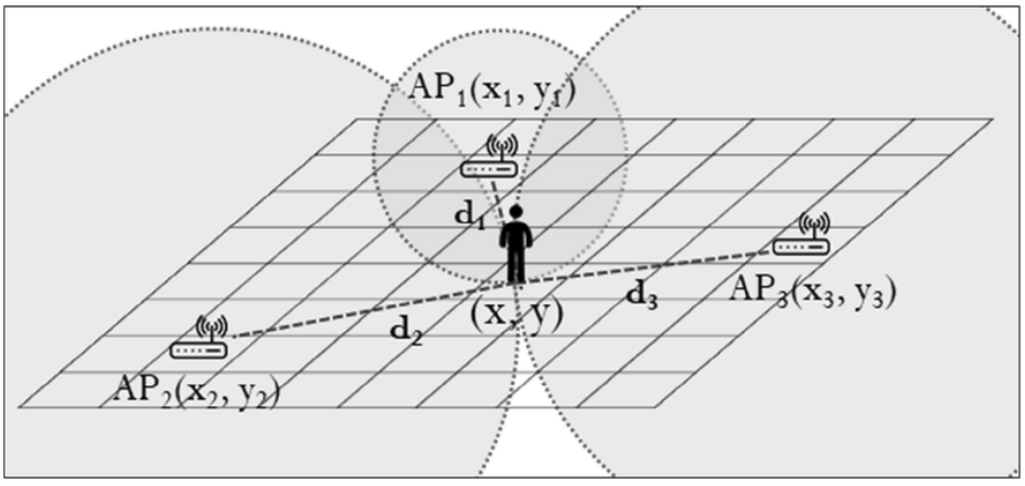


Figure 2.1: Overview of Triangulation technique using 3 access point stations. d_n is the distance of the user at (x, y) to corresponding access point $AP_n(x_n, y_n)$. Image source: [1]

Triangulation is a technique that computes the positions of a receiver based on the measured angles between two known points. From those angles, the distances are computed which are used to calculate coordinates for the target points.

Bahl et al. [31] presented one of the first pieces of work that aimed to branch away from energy-consuming GPS and instead use the RF LAN technology and signals in the 2.4 GHz frequency to put forward a WiFi based technique for localisation. Using this technology, Bahl hoped to triangulate a user's co-ordinates for localisation using just signal strength and performed tests using 3 base Access Point stations (2.4 GHz, 2 mbps) in open hallways and closed locations around a building. The users were give laptops running Windows 95, which they had to carry around with them. By doing this, Bahl used each base station to collect signal strength information from each laptop device, including user's orientation, which was used to help distinguish between variations in signal strength readings from users. During these tests, users would confirm their co-ordinates by selecting a position on a map, and then a closest match search would be initiated using multi-dimensional searches (R-Tree, K-NN ect.). From the closest match, distance is compared using the Euclidean distance metric. The results show that Bahl can track moving users with an error distance of 3.5 meters, 19% worse than a stationary user, but with an overall median resolution of 2-3 meters.

The work in [33] built a light weight and high accuracy localization system for off-the-shelf smartphones, by using WiFi and microphone sensors. When the user is determined to be within Line-of-Sight of at least three beacon devices, the authors apply the triangulation technique to estimate the user's actual location. However, they note that "in scenarios where the user is moving, (e.g. walking or running), this triangulation technique is not always practically applicable due to the very short duration available for location computations".

2.1.4 Propagation-based Algorithms

Propagation based algorithms [38, 39] provide an estimation of a user's position by measuring the signal strength and taking into consideration the path loss from radio towers or WiFi transmitters [40, 41]. The Multipath effect [42] introduces random variations in the received signal amplitude over a frequency bandwidth. This effect also varies depending on the location of the antenna as well as the type of antenna used. Modelling the radio propagation is not an easy task in an indoor environment because of the probability that path loss will remain a constant; path

loss may change depending on the floor layout, moving objects, human density ect. Radio propagation in such environments suffer from multi-path effect, and therefore the accuracy of estimated location would be decreased. Alternatively, an approach [43] is to estimate the distance by using the attenuation of emitted signal strength, to attempt to calculate the signal path loss.

The use of RSSI for distance estimation is inconsistent and volatile to environmental conditions [44]. Instead, a common technique is to use WiFi fingerprints, which are represented as a vector set of sampled WiFi RSSI values from all the visible AP from around the smartphone device, generating signal strength maps to be used for matching against fingerprints from other smartphone devices.

2.1.5 Fingerprinting Approach

Since the work presented in [45], there has been huge advancement in the field of smartphone technologies, which means that users can be tracked by much smaller devices that can fit into their pockets. Most modern work is done using smartphones [46–49], or from within the Access Points themselves [34]. Due to the infeasibility of using just signal strength model-based distance estimation, most modern work now employs a WiFi fingerprinting-based approach, which provides more accuracy, is more robust and is ultimately cost-effective for indoor environments.

Location-fingerprinting (LF), mainly consists of two phases [2]: the first is an offline phase, which is then followed by an online phase. The offline phase searches for discoverable WiFi access points and collects their IEEE 802.l1b WiFi signal strength for a training database (see figure 2.2); in this case, the trained database is a collection of rooms, enrolled by a snapshot of their location fingerprint vectors. In the second online phase, the fingerprints are retrieved by the smartphone device and the location is estimated by matching similarity or distance to the LF database of enrolled rooms (see figure 2.3). For social interaction detection, this method can be adapted by modifying the offline phase to collect WiFi signal strength data from every smartphone device, and then in the online phase, device proximity can be detected by matching each smartphone's recorded fingerprint against the data collected by all other smartphones.

Location	RSS(in dBm)			
	AP 1	AP 2	AP 3	AP 4
Room 1	**-90**	**-70**	**-60**	**-50**
Room 2	**-70**	**-90**	**-55**	**-70**
Room 3	**-60**	**-50**	**-85**	**-50**
Room 4	**-40**	**-50**	**-60**	**-90**

Figure 2.2: Overview of the offline phase, which collects WiFi RSS data to enrol rooms via the generate of a Location Fingerprint. In this image there are 4 access points, APn, and each room has a different signal strength to each access point. Image source: [2]

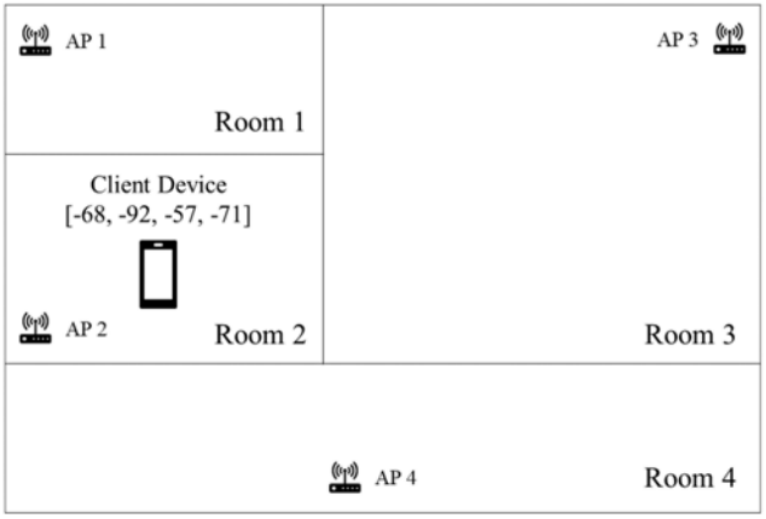


Figure 2.3: Overview of the online phase, which attempts to locate users based on the WiFi fingerprint collected by the smartphone device. In this image, the fingerprint collected is quite similar to the enrolled fingerprint of Room 2. Image source: [2]

Work done in [49] by Farshad et al. looks into the various aspects underlying a WiFi fingerprinting system, specifically, an investigation into the different definitions for fingerprinting and location estimation algorithms across different indoor environments. They compared the location estimation algorithms and modify WiFi fingerprints based on patterns, such as WiFi strength, stability, variance, constancy and coverage. This involves taking the raw WiFi signals and creating a subset in accordance to a particular criterion. Each subset will contain 5 vector entries, however, Hu et al. mentions in [50] that these approaches can either introduce some false information or omit some useful information. Therefore, it can be presumed that introducing new information into the fingerprints can cause dilution of vectors, and removing

information can eliminate some of the more important data aspects from the analytics process.

Farshad et al. [49] looked into how the WiFi Fingerprinting system worked within an office environment and a shopping centre environment by calculating location estimation errors of all possible combinations of location estimation algorithms. These include deterministic distance metrics such as Euclidean distance, Manhattan distance, Mahalanobis distance [51] and probabilistic techniques such as Gaussian distribution and Log-normal distribution. It was discovered that for the office environment, 'default' (unmodified fingerprint vectors) performed with the poorest accuracy, and the best accuracy was achieved through 'Manhattan distance'. However, in the shopping centre environment best results were achieved with 'Mahalanobis distance' in partnership with 'constancy' subset of WiFi fingerprints. This finding suggests that signal strength needs to be processed in a specific ways for different environments, in order to produce better accuracy dynamically.

2.1.6 WiFi Probes Based Approach

Although WiFi fingerprinting yields good results, a limitation is that all devices must install the same app which collects the WiFi data. Another technique is to modify the firmware of WiFi access points and listen to the packets of connected smartphone devices (probes). Instead of the smartphone device collecting all of the information about nearby Access Points, the Access Points collect all of the information about nearby devices (as outlined in the discovery section of figure 2.4.

Vanderhulst et al. [34] presented the implementation of a human encounter detection framework for measuring and analysing human behaviour in social settings. This framework was evaluated through controlled experiments and accompanied by live deployment, however, they concluded that these were limited in number of participants and device variety. In this paper, Vanderhulst et al. propose to use WiFi probes which are radiated from mobile devices periodically, as well as existing WiFi access points to automatically capture radio signals, meaning that there is no need to use any sensors from the user's smartphone device, such as audio/video signals, GPS coordinates, accelerometer or WiFi scans, and all of the data collection is performed

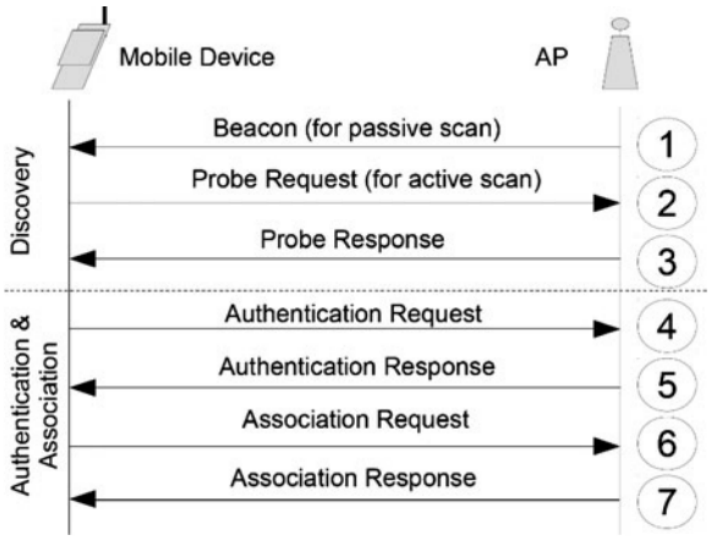


Figure 2.4: IEEE 802.11 Network Discovery and Association Process overview. The WiFi Probe scanning overview can be seen in the 'discovery' section. Image source: [3]

within the access points themselves.

In their experiments, Vanderhulst et al. used 4 WiFi monitoring nodes (Meshliums) which were deployed in an open-plan office space shared by 50 people to detect whether people were queuing or not, as a test for volatility in groups. Each nodes was located 10-20 meters apart and Received Signal Strength (RSS) fluctuations between devices were aggregated by multiple detection samples over a 1 minute window. They acknowledged that their system was not detecting any encounters for the queuing scenarios. They also found that their system could detect encounters in the workplace scenario, but with a smaller window size, because they could not distinguish between individuals joining, or departing the workspace, unless the encounter was longer than what they consider to be the boundary of an encounter.

In [35], Wang et al. used WiFi probes to determine the occupancy of buildings via a novel Markov based feedback recurrent neural network algorithm and by diagnosing the captured connection requests, responses and RSS values. Wang et al. concluded that WiFi probe-based occupancy models have lower prediction error in terms of occupants count.

In general, single modality methodologies, such as WiFi fingerprinting or probe-

based approaches, the RSSI value can be used to determine the proximity between smartphone users within a certain accuracy, but proximity is not enough information to be making a presumption about whether or not an interaction is taking place, since people can be close but within different social groups, queuing separately or in the same work environment.

2.1.7 Portable Hot Spot Approach

Osmani et al. [36] proposed 'Comm2Sense' to exploit sensors embedded in smartphone devices and to detect the proximity between two or more individuals with a median accuracy of 0.5m. This distance estimation was calculated by allowing all devices to act as a WiFi transmitter and receiver, in an effort to trade off the time-consuming technique of WiFi fingerprinting, however, user effort was required to calibrate phone signal. The work was carried out with the consideration that smartphones may one day have an implementation of WiFi direct, which was added into Android 4.0 as "WiFi Peer-to-Peer".

The testing of this system was performed in three environments: an office, a balcony and a meeting room. RSSI values were measured for 5 minute intervals at increments of 0.5m, to the point that the signal strength had degraded to a minimal level. Osmani et al. set the transmission power to 0 dBm and manually controlled the state of the portable hot spot, however, this required the installation of a custom firmware "Cyanogenmod", which can only be done on devices which allow OEM unlocking [52] (rooting the devices). The results suggested that this method would require a training phase for every phone model, however, they attempted to address this by using a path-loss model. By doing so, the system yielded better accuracy in line-of-sight, but was sensitive to reflections and multiple-path propagation.

2.1.8 Ultrasound

Indoor localisation has received an increasing amount of attention primarily due to it's ability to leverage Internet of Things and ubiquitous connectivity. Various techniques have been used to determine device localisation within a building [53, 54], for example, in [54] Hazas et al. created a calculation for the relative position of smartphone devices by relying on custom hardware that can emit ultrasonic signals.

This unique technique breaks away from common methods which use samples of WiFi or Bluetooth signals. The technique was proven to be very accurate, with a 50th percentile error of 2-4cm, however, a system like this needs ultrasound emitters and receivers, which are currently not available in any smartphone device.

2.1.9 Bluetooth

Wireless technologies such as Bluetooth are generally the mechanisms which have been proposed to provide indoor localisation services for users due to popular use of off-the-shelf smartphone devices which come equipped with the technology, their potential for high accuracy, and a relatively low infrastructure cost. Moreover, Bluetooth-based localisation has gained prominence due to the introduction of Bluetooth Low Energy (BLE) which achieves a much lower connectivity latency and a power reduction of nearly 50% when compared to Bluetooth 4.0 and below. For Bluetooth, discoverability signals are typically transmitted by one device and then captured and fingerprinted by a receiver device to infer proximity between the two points.

Comparatively, Bluetooth uses less power than WiFi [55] and is therefore more viable for indoor location services, however, WiFi is more widely available in urban areas; Bluetooth devices would need to be installed as an addition to existing building infrastructures which can be undesirable.

Lots of work involving Bluetooth technologies involve proximity based approaches which look to utilise the smartphones onboard sensor and establish the devices distance from other Bluetooth transmitters [37, 56, 57].

2.1.10 Proximity-based Approaches

There have been many ways of detecting proximity between devices using wireless sensors to estimate whether two participants are close enough for a social interaction to be feasible, with Bluetooth being a main instrumentation. Many techniques rely on the Time or Angle of Arrival, and measuring the Received Signal Strength (RSS) [58] to estimate the distance by comparing the power constant that indicates the signal strength in decibel-milliwatts (dBm) at a known distance [59, 60]. An advantage with using BLE devices for proximity estimation is that manufacturers

can configure the Bluetooth transmitters to use their own calibrated power constant, which provides better proximity estimation.

Work in [61] uses WiFi signals and the captured RSSI values to determine proximity between smartphone devices. They also use Bluetooth signals to capture ground truth of the interactions, however, they conclude that due to the imperfect firmware and software running on the smartphones, Bluetooth data is not always available; not all users are scanning and discoverable at all times. This can introduce a situation in which two persons are proximate, but Bluetooth does not capture those types of events.

2.1.11 Improving Accuracy when using RSS Signals

Bluetooth, although low powered, typically suffers from noisy signals. In [62], Röbesaat et al. describes an indoor location-based system which attempts to smooth the noise present in Bluetooth RSSI signals by applying a Kalman Filter [63], and to determine position fixes from combining trilateration and dead reckoning fixes to establish if a user is close to a fixed location. In this work, Röbesaat et al. obtains an accuracy of less than one meter and high stability position fixes when using the Kalman filter. Considering that this approach is applied directly to the signal strength values received, the Kalman smoothing technique can also be applied to WiFi signals to improve signal strength stability in diverse and noisy environments [64].

2.2 Multiple Modalities

Multiple modality is a technique used to combine different types of sensor technologies in order to achieve a more robust system [65], to improve security [66] and to perform with better recognition accuracy [67]. Researchers typically combine data sets in order to improve the robustness of their systems. These data sets are usually generated by the data collected from other sensors, such as Bluetooth, WiFi, audio and magnetometers.

Approach	Modalities	Target	User Interface	Performance
Meeting Mediator [68]	WiFi Bluetooth Accelerometer Infrared sensor	Wearables	App	76% dominent speaker detection
DARSIS [69]	Bluetooth Accelerometer Magnetometer	Smartphones	N/A	81.40% accuracy
Matic et al. [70]	Microphone Accelerometer WiFi	Smartphones	None	89% accuracy
Katevas et al. [71]	Bluetooth Accelerometer Gyroscope	Smartphones	None	77.8% precision 86.5% recall
Opo [4]	Bluetooth RFID Ultrasonic	Wearables	N/A	Interaction detection accuracy of 5cm

Table 2.2: Comparison of reviewed multi-modal proximity/distance estimation-based systems

2.2.1 Bluetooth + WiFi

In Meeting Mediator (MM), Kim et al. [68] created a portable context-aware computing system which can detect social interactions to enhance group collaboration. MM attempted to bridge the gap amongst group distribution by detecting social interactions between group members and then providing a categorisation of the behavioural differences between dominant and less-dominant group members. The publication described a study, which aimed to answer the questions: How can we assist groups to be more effective? Why do distributed groups and co-located groups perform differently? What impact does dominant behaviour have on group dynamics and performance? The system worked by having participants wear Sociometric badges around their necks to capture the user's voice, body movement through accelerometer, Bluetooth proximity data between participants, face-to-face alignment through infrared, and WiFi signals from fixed based stations. It concluded that their system, was able to quantify the characteristics of dominant people and their influence on other people within groups, which then allowed it to change distributed groups so that they "...collaborated more like co-located groups, and reduced the difference between dominant and non-dominant people by making everyone more

energetic and involved". The results showed that MM was able to correctly identify 76% of the people who were perceived to be "dominant" by themselves, through the use of multiple modality (Bluetooth, WiFi and accelerometer). This is also a prime example of the importance of having technologies that can be collect information about the social interactions of individuals; when you consider that smartphones are able to manage group distribution by sorting members in accordance to dominance, you can also then envision the potential to explore other areas, such as tracking personality traits, and then recommending friends based on a categorisation of these characteristics.

2.2.2 Accelerometer

Work in [69] combines Bluetooth with "uDirect" [72] - an algorithm that can determine the standing orientation of the smartphone user based on accelerometer and magnetometer data. The detection of the orientation of users can allow the discovery of whether participants are facing each other as indication of interaction. However, the particular technique relies on the proper placement of smartphones during social interactions.

In [70] Matic et al. builds a system which captures speech activity from an off-the-shelf accelerometer that rests on the participant's chest and detects the vibrations that are generated by vocal chords during phonation.

Work in [71] focuses on a machine-learning-based approach which leverages Bluetooth (BLE), accelerometer and gyroscope technologies to predict social interactions for a variable-sized crowd by relying on the data collected by smartphone devices. The system achieves a 77.8% precision and 86.5% recall; notably a 30.2% performance increase when compared to a Normalized Proximity based approach captured during their own experiments.

2.2.3 Using RFID Devices

A section of the literature is also reserved to the capturing of face-to-face interactions which occur between participants. Researchers believe that they can capture these more accurately by the use of wearable technologies such as badges or smart watches.

The work in [4] develops "Opo" to capture social interactions without the need for

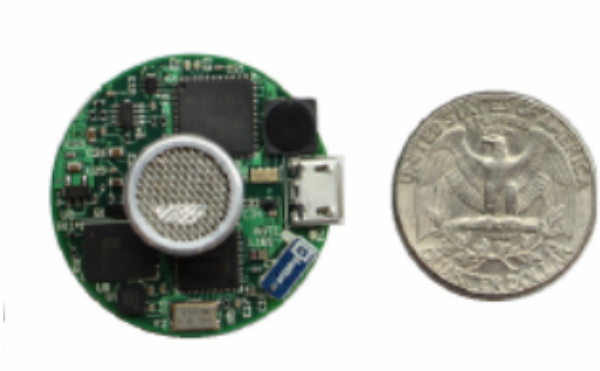


Figure 2.5: The Opo wearable, next to a coin for scale. This device is typically worn by participants on the front-facing part of their bodies in order to capture the face-to-face interactions that participants may experience. Image source: [4]

infrastructure nodes or RF discovery protocols by using a wearable sensor with a ultrasonic wake-up radio. The sensors are the size of a large lapel pin (see figure 2.5) and keep a 4-day lifetime with a 40 mAh battery. Opo is typically worn by participants on the front-facing part of their bodies in order to capture the face-to-face interactions that participants may experience (see figure 2.6). Opo claims to track interactions with a ranging accuracy of 5 cm.

In [73] the authors present two studies to assess the construct and criterion validity of RFID badges used to measure social interactions. These RFID badges were worn by participants in multiple experiments during orchestrated social settings. They concluded that RFID badges are in part a valid measure of social interactions and the captured RFID signals are likely to be a true face-to-face interaction.

2.3 Social Sensing

Social sensing is rapidly progressing as a pervasive sensing paradigm, where people are used as sensors to obtain situational data about the physical world. This data is collected either directly from human observers, or by crowd-sourcing using sensors in smartphones or various other wearables (e.g., Smartwatches, fitness bands). This process typically involves a massive amount of sensory data. Social Sensing approaches new challenges in behaviour science by being able to curate new types of data sets about individuals and their surroundings.

Figure 2.6: The Opo wearable, being worn by participants. The device simply clips onto a tie or a pocket in a discrete manner. Image source: [4]

Smart cities aims to improve sustainability, create economic development, and enhance quality of life factors for people living and working in the city. In this area, social sensing focused on challenges such as preserving privacy of participants [74], improving the energy efficiency of sensing devices [75], and the CarTel system [76] aimed to monitor traffic patterns in a city to help drivers avoid congestion areas.

In the space of assisting people with disabilities, [77], Halperin et al. used a WiFi signal based system to help blind users determine the presence of one or more people in a room. In this work, the approximate distance of the users was determined by reading the strength of the nearby WiFi signals emitted from nearby access points.

Unlike physical health conditions, the treatment and monitoring of mental illnesses can rely on subjective measurement. This is where social sensing can become helpful. In [78] Jiang et al. designed a portable wearable device for capturing data on long-term personal mental health and well-being. This was implemented using multiple

sensors, including a MEMS microphone, a behaviour sensor (using motion tracking with a gyroscope/accelerometer chipset) and environmental sensing (temperature and humidity sensor). In this work, the authors find a correlation of social features and questionnaire scores.

By curating new types of data sets about individuals and their surroundings, social sensing ultimately aims to assist people and improve their well-being using technology.

2.4 Speech Processing

Since most social interactions contain speech between participants, it makes sense to use audio recordings as a technique for conversational detection. Audio is an important part of social interaction between individuals. Some work uses the on-board microphones of smartphones to record audio and use it for speaker recognition or conversational turns [22, 79–81].

2.4.1 Voice Activity Detection

Voice Activity Detection (VAD) is used to detect when speech occurs within an audio stream, which can then be used to strip out all of the background noise from an audio sample [82]. This technique is widely researched and used in audio signal processing systems for speech encoding, speech recognition, and Speaker Identification tasks [83], to eliminate some of the background noise and to improve the performance of the system. Other work focuses more on the retrieval of spoken content from audio samples, using Automatic Speech Recognition (ASR) technology and aims to generates text transcripts of what the user is saying [84].

2.4.2 Speaker Identification

Speaker Identification is the process of identifying a speaker from data which represents the individual's speech biometrics [5]. The characteristics that define a person's voice are different and the challenge is to differentiate speakers from other speakers. Many systems use Speaker Recognition as a verification method, to enrol new speakers into a data set or for security - to verify that the person is who they say they are [85]. Other work use Speaker Identification techniques to count the

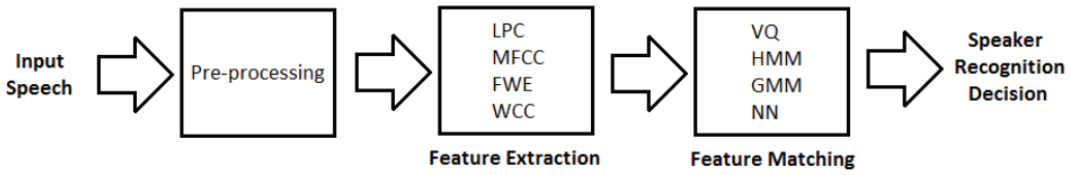


Figure 2.7: Outline of typical speaker identification systems. The input speech is taken and pre-processed, and then features are extracted. These features are passed into a classification stage which can be used to make a decision on which participant the input speech belongs to. Image source: [5]

number of speakers from a session of audio data [22, 81, 86].

The work done in DSP.Ear [22] presents a smartphone system that explores the use of the latest low-powered Digital Signal Processing (DSP) co-processor technology in mobile devices to enable the continuous and simultaneous operations by offloading tasks to the DSP instead of running them on the CPU. DSP.Ear classifies speakers from speech signals and also estimates the number of speakers in a room via an unsupervised speaker counting technique presented in the Crowd++ paper [81], which uses a voice activity detection technique that requires the use of pitch, specifically the YIN fundamental frequency estimator [87]. Crowd++ counts the number of speakers by comparing MFCC features extracted from speech and boosting accuracy via a gender estimation produced by the pitch of the speech. DSP.Ear uses 10-minute speech samples recorded by 22 speakers from their University department. The Speaker Identification reuses the algorithmic elements introduced in the emotion classification from [80]. A 128-component background *Gaussian Mixture Model* representative of all available speakers is built and MAP adaptation is performed on *Perceptual Linear Prediction* features [88] from speaker specific utterances to obtain the speaker-dependent models. The Speaker Identification technique performs at 95% accuracy across the 22 participants.

The speaker counting in DSP.Ear is applied when the conversation is over, by extracting MFCC features into clusters. The final stage of their algorithm compares the clusters against each other using the cosine similarity [89] and the inferred gender of the speaker represented by the cluster. The total number of identified clusters is inferred as the number of speakers in the conversation.

2.4.3 Using Vector Quantization

Vector Quantization (VQ) is a popular unsupervised learning algorithm for speech features. It allows the modelling of probability density functions by the distribution of prototype vectors by dividing vectors into groups that have the same number of points closest to them. In [90] Mel frequency Cepstral Coefficients (MFCC) are extracted from the voice of 100 speakers and the Vector quantization technique is used to identify each speaker, with 82% precision. Importantly, a finding from this study explains that as the number of speakers increase, the number of errors increase because the of distance to each centroid of each speaker can be similar to another. A speaker identification system needs to be scalable, otherwise there are limitations on the ability to distinguish between diverse voices from many speakers.

2.4.4 Using Probabilistic Neural Networks

Probabilistic Neural Networks are similar to a typical Back Propagation Neural Network [91], except the sigmoid function is replaced by an exponential function, which provides the advantage of increased learning speed computation over other types of architectures. The work in [6] makes use of a Probabilistic Neural Network to classify speakers. This work extracts MFCCs and their Deltas [92] from the Mel spaced Gaussian Filter Banks and yields 94% accuracy across 5 speakers.

2.4.5 Using Deep Learning

Speech processing plays an important role in any speech system. Using Deep Learning has shown to produce promising results. This section covers the use of Deep Learning for supervised and unsupervised techniques.

2.4.5.1 Supervised Techniques

Work produced in VoxCeleb [7] uses video data from YouTube to extract speech from people speaking in interviews, typically from popular celebrities. Their workflow results in a database where high-confidence video segments corresponding to the target speakers are automatically annotated. In their experiments, they use this data to train for speaker identification. Their Supervised 2D Convolution Neural Network provides a model for closed-set identification of 1251 speakers, and showed

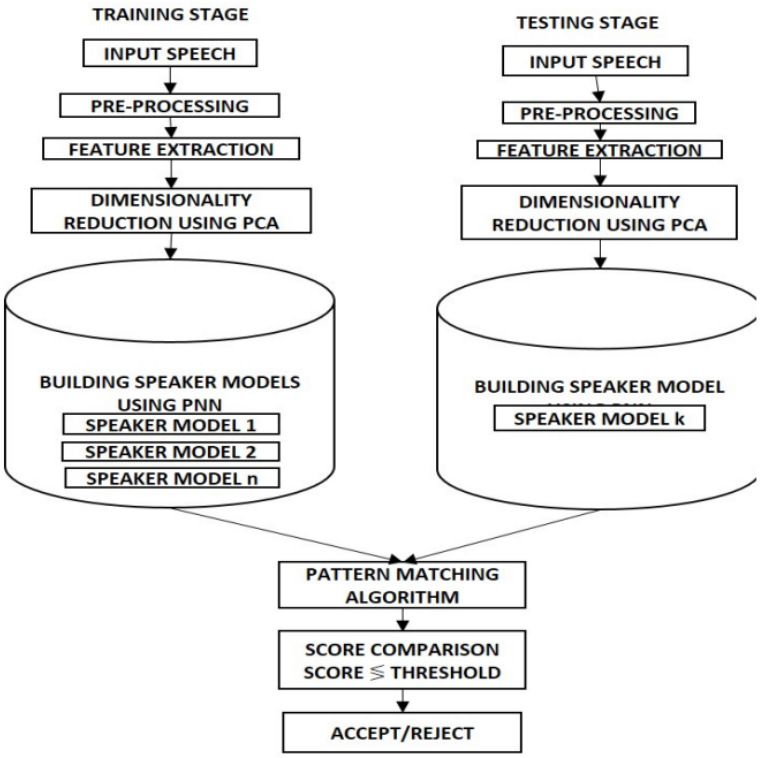


Figure 2.8: Outline of a typical Probabilistic Neural Network speaker identification system. Image source: [6]

that a speaker identification model trained on such data achieves 80.5% accuracy.

In [93] the authors present a fully supervised speaker diarization approach, using unbounded interleaved-state recurrent neural networks (UIS-RNN). Speaker Diarization is the process of partitioning an input audio stream into homogeneous segments according to the speaker identity, and is used to separate overlapping speakers in an audio segment. Their work performs as low as 7.6% Diarization Error Rate by using a trainable unbounded interleaved-state Recurrent Neural Network.

2.4.5.2 Unsupervised Techniques

An autoencoder is a transformation applied to an input vector that tries to map samples from one space into the same space by trying to squeeze the high dimensional space through a lower dimensional bottleneck (known as the encoder). The autoencoder can then reconstruct the original samples (known as the decoder) and a loss function can calculate the error of the reconstruction. In unsupervised speech processing, the output of a deep learning model is not of utmost importance; we

instead care more about the hidden representation that the autoencoder has learned.

The work in [94] uses WaveNet [95] autoencoders to extract meaningful latent representations of speech. Their method of Vector Quantised-Variational autoencoder preserved the most phonetic information from speech, whilst also being the most speaker-invariant.

In [96] the authors present a clustering approach for fully unsupervised Voice Activity Detection. Their work showed good results when compared to the two-component GMM baseline by leveraging a Hartigan dip test [97] in a recursive strategy to segment the feature space into prominent modes.

2.5 Limitations of State of the Art

In social sensing, sensors are fundamental elements, where accuracy and stability are of great importance [55]. Any system needs to be scalable, and be able to handle a large number of users in a large space. Scalability is a major challenge to complex sensing systems. In addition, smartphone devices are battery-operated and therefore require apps to be developed as lightweight as possible [98] in attempts to increase lifetime. A common approach to reduce energy is to suspend sensing for periods, called duty cycling [36, 99], however, suspending communication or sensing could cause the smartphone to miss important events. Therefore, the common challenges in this domain would be to maximise the accuracy of detecting significant life events, whilst at the same time taking into consideration scalability and battery performance.

The gap here is the proximity/location techniques which make the assumption of social interaction just because people are close. In many common scenarios, this is not the case, for example when people are queuing but not a part of the same social group or interaction, travelling together on public transport, and sitting on tables near each other in a restaurant or cafe. Voice techniques typically address the problem, however, most speaker identification techniques require training with a speaker database. This fact imposes limitations on the potential of wide deployment of such systems considering the burden of collecting voice samples from participating users.

Clearly, there is a need for systems that do not require special infrastructure, can capture social interactions in challenging scenarios and can scale easily over a large user base.

CHAPTER 3

Capturing Social Interactions using WiFi and Audio Signals in Smartphone Devices

3.1 Introduction

Social interactions represent a significant part of our daily lives. They are considered a significant aspect of the quality of people's daily lives [100], as well as an important activity that enables collaboration and creativity [101]. In recent years there has been an increasing interest in developing technologies that can capture the social behaviour of people. Within working environments, analysis of the social behaviour of employees has been shown to reflect the performance and productivity of teams [102]. Within the health and well-being domain, long-term tracking of social behaviour has been used as indicator for changes in mental health and perceived quality of life [103].

People-centric sensing technologies have been employed in a number of scenarios to develop systems that can passively capture social interactions. Wearable and mobile devices (e.g. smartphones) have been used to infer social behaviour by analysing the mobility patterns of individuals [80]. Traditionally, most of the approaches

that have been used in such scenarios assume that proximity between individuals is an indicator of social interaction. This may be a valid assumption for certain situations (e.g. people participating in a meeting), but the assumption may not hold when considering situations where social interactions take place within crowded environments, involving multiple social groups. Such scenarios are very common in the daily lives of people. Having a chat in a crowded café, or interacting with different people during a networking session in a conference, are common situations where proximity may not be sufficient to correctly identify the people involved in an interaction.

In this work, we attempt to develop a system that can accurately capture social interactions within challenging scenarios where multiple social groups interact within close proximity of each other. In order to distinguish the closely located social groups, we rely on analysis of audio captured by the smartphones of the participants and aim to identify which social group they participate in. Our hypothesis is that the sound patterns captured by the smartphones of the people participating in a conversation is sufficiently different from the sound patterns of people not involved in that interaction event, or participating in a different social interaction even if the groups are within a few meters from each other. Intuitively, we consider that people participating in a conversation that takes place in a noisy or crowded environment have the tendency to raise their voices enough to be heard by the people involved in the conversation. This natural behaviour is enough to produce distinct sound patterns that are very similar for the people participating in the conversation, and sufficiently different from the sound patterns captured by the smartphones of people in near-by social groups. We demonstrate the design of a system that relies on a combination of WiFi fingerprinting and Audio fingerprinting captured by smartphones that are in the participants' pockets, or on tables in front of them. Through a range of controlled experiments and a real-world deployment in a noisy café, we demonstrate that the system can achieve an average precision of 88%, while maintain a power consumption of less than 3% of the battery life per day.

3.2 Related Work

Traditional techniques in passively capturing social interactions involves the use of RF technologies as a means of capturing the co-location of users. Examples include systems that use Bluetooth on smartphones [69, 104] or specialised wearable RF tags [105, 106] to detect face-to-face proximity. Although such techniques can offer an approximation of the social behaviour of users, ultimately co-location does not always imply social interaction.

WiFi fingerprinting has been widely used as a way of localising users within buildings [107, 108]. As such, the technique has also been used to detect co-location between users in environments where sufficient WiFi infrastructure is present [109]. However, such techniques suffer from similar limitations to other proximity based approaches where the co-location estimation does not imply social interaction. Recently, there has been significant work on the capture of social interaction passively through the collection of WiFi traces of users' smartphones using the WiFi infrastructure of a building [110, 111]. These techniques allow the tracking of social interaction without the need for the users to install a particular application on their phone. They allow the passive tracking of large numbers of users, but require access to the WiFi infrastructure of a given environment. In practical terms, these techniques can only be employed in certain environments, and do not allow the capture of social interactions throughout the daily lives of participants. Moreover, the passive tracking of smartphones without the need for an app installation raises significant privacy issues. Smartphone OS such as iOS have recently employed MAC obfuscation techniques to avoid such passive tracking thus rendering these approaches infeasible.

Since most social interactions contain speech between participants, it makes sense to use audio recordings as a technique for conversational detection. Some work uses the on-board microphones to record audio and use it for speaker recognition or conversational turns [22, 79–81]. Other work uses audio for indoor localisation [112] or proximity detection [113]. The work done in DSP.Ear [22] presents a smartphone system that extracts emotions from speech signals, estimates the number of speakers in a room [81], and detects the identity of speakers and identifies common ambient

sounds. This type of work relies heavily on the use of machine learning techniques, requiring training of the voice recognition component through appropriate sound datasets, often involving sound samples by the user. In many scenarios, such an approach would be infeasible for large scale deployment limiting the applicability of the approach.

3.3 Motivation

Current approaches in capturing social interactions tend to rely on secondary signals such as co-location, as proxy for an actual social interaction. Indeed, when individuals are close to each other there is a high probability that they are interacting with each other. However, there are numerous scenarios where such approaches can lead to erroneous results. People working in shared office environments may be co-located but not interacting; when interacting with people in busy places, such as a restaurant or a social event, co-location may involve more people than somebody is interacting with. In order to enable a more accurate detection of social interaction, there is a need to move beyond co-location.

Audio has been shown to be a more accurate indicator of actual social interaction, as a means of capturing the actual conversations of people involved. However, relying on heavy-weight speech recognition or speaker recognition requires personalised voice training [80]. Such approaches do not scale well, as machine learning algorithms need to be trained with voice sample of participants.

In this work we aim to develop a system entitled "Next2Me" (see overview of the system architecture in figure 3.1) that combines co-location and audio sensing to accurately detect social interactions in challenging environments. Such environments involve close co-location of social groups, interacting within busy environments. In our approach we do not require prior training of the system with audio samples, neither from the users or the environment. Instead, we rely on the comparison between sound signals captured by the users' phones, as indicators of close proximity. Our motivation is based on the assumption that sound signals will have unique patterns for the people participating in a conversation, and are different from those

in nearby conversations. Even in a noisy environment, people tend to talk louder to make sure that their conversation is heard by the participants from within the same group. Intuitively, we expect that sound samples captured by smartphones within a social interaction will have similar sound patterns, containing primarily the voices of the people participating. In our overall system, we utilise co-location as a means to trigger audio sensing when there is a high probability of social interaction. Sparsely sensed audio samples are then used to formulate a *"sound fingerprint"*. Comparisons between sound fingerprints are then used to discover the social networks of co-located users. The proposed approach does not require any special infrastructure, and can be used in any environment where there are sufficient WiFi signals to facilitate co-location sensing.

We orchestrated various experiments using participants to capture real data and to determine the performance of our proposed system. Experiments 1 and 2 were conducted in our Preliminary Study section, and experiments 2 and 3 were analysed in our Evaluation section. Experiment 4 acted as a real-world scenario experiment. Please refer to table 3.1 as an outline for all of the experiments covered in this chapter.

Experiment	**Participants**	**Scenario**	**Groups**	**Devices on Table**	**Devices in Pockets**
1	8	Meeting	1	8	0
2	6	Co-working	2	6	2
3	7	Conference	3, 2, 3	0	7
4	6	Coffee	2	6	2

Table 3.1: Outline of the various experiments conducted using the proposed Next2Me system

3.4 Preliminary Study

In order to develop a robust system for capturing social interactions using smartphones, we initially attempted to explore the extent to which WiFi based proximity detection can enable the identification of such interactions. Furthermore, we also tried to explore how audio signals can be analysed to further assist in identifying social interactions. Our aim was to explore whether the combination of these two

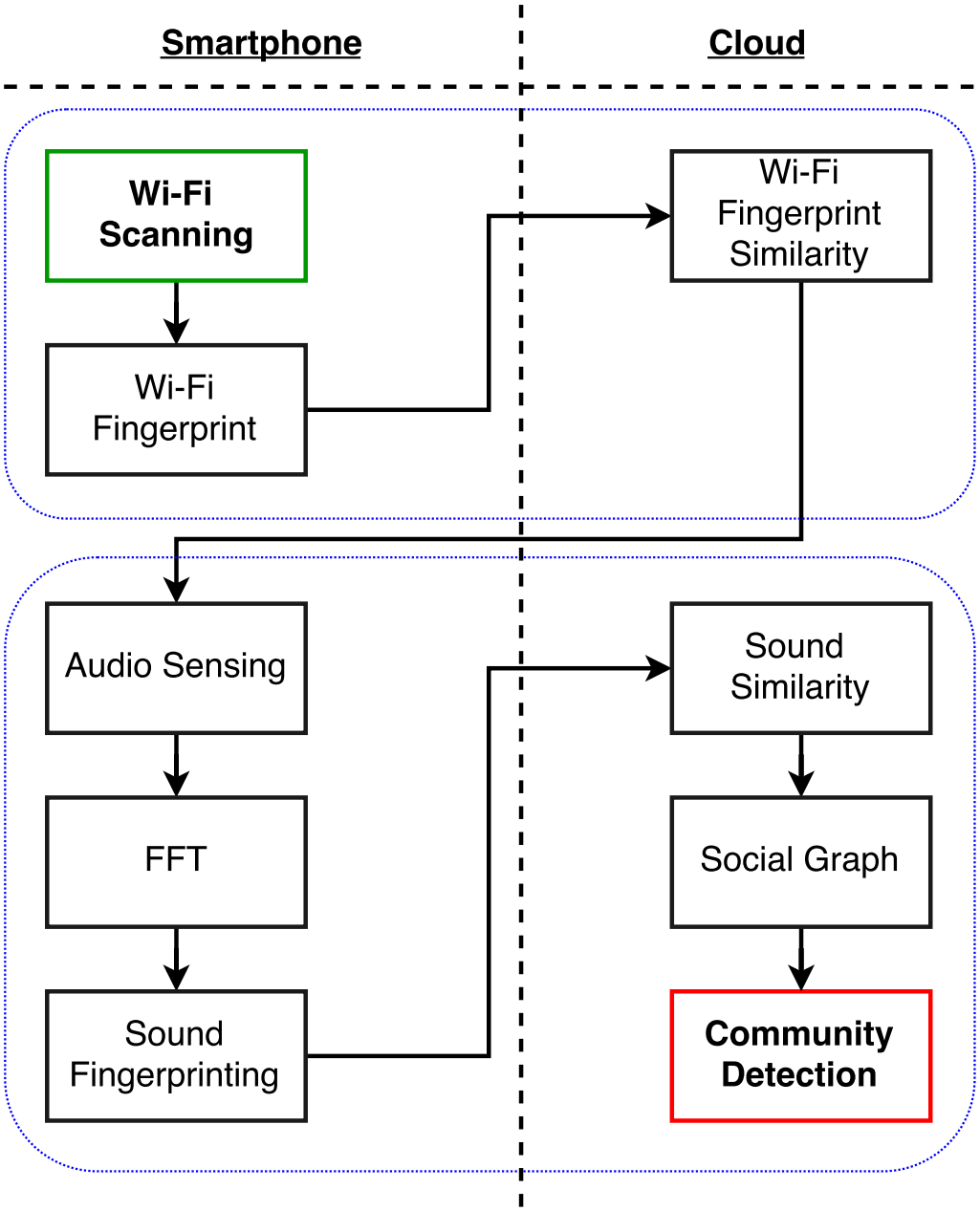


Figure 3.1: Architecture overview of the entire Next2Me system. Firstly, WiFi signals are captured and transformed into WiFi fingerprints until two or more devices are deemed co-located due to the similarity of these signals. Then, when co-located, these devices can start capturing audio signals. The audio is transformed by an FFT into a Sound Fingerprint, which are then compared and split into communities.

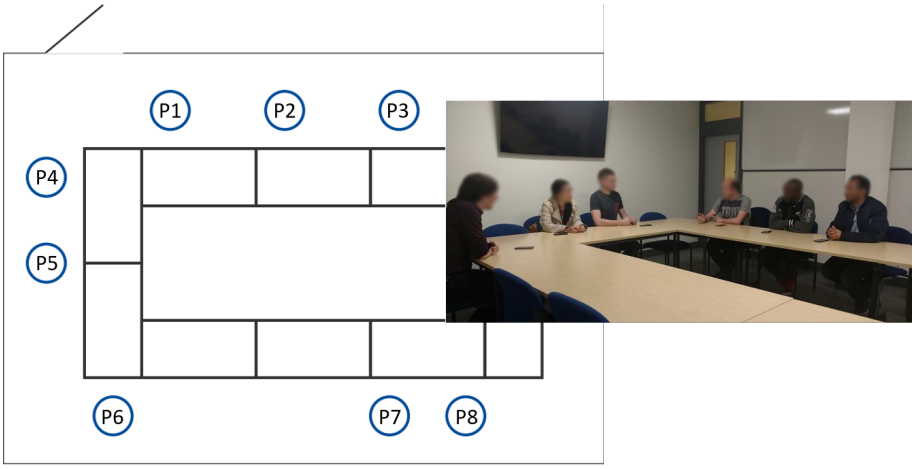


Figure 3.2: Set up of the meeting scenario experiment (Experiment 1). Left: an overhead view of the table setup where P_n is the seating position of a participant

modalities can lead to a robust social sensing system that does not require prior training.

For the preliminary study, we developed a data collection app for Android smartphones. The app was running continuously, capturing WiFi and audio data in the background. Specifically, the app scanned for available WiFi access points every 10 seconds, and recorded the MAC addresses of the access points and the RSSI value of the signal strength received by each access point. At the same time, the application recorded audio continuously for the duration of the experiment.

3.4.1 Experimental Setup

The call for participants involved sending out an email to the entire Department for Engineering and Digital Arts (EDA) at the University of Kent. The 19 participants of these experiments were personally invited to join the study. The participants were all students at EDA building at University of Kent. Users downloaded the APK file for Android devices and installed it onto a smartphone device, which was provided for them. This APK was uploaded to the Google Play Store, for ease of access.

To carry out the experiments, we submitted an ethics application to the University of Kent Faculties Support Office. It was deemed that our project did not require ethical approval, and we were able to proceed with the research (reference number 0151617).

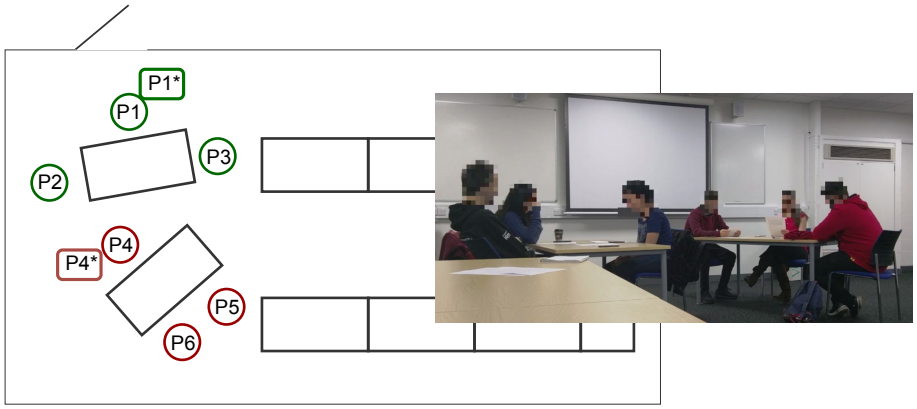
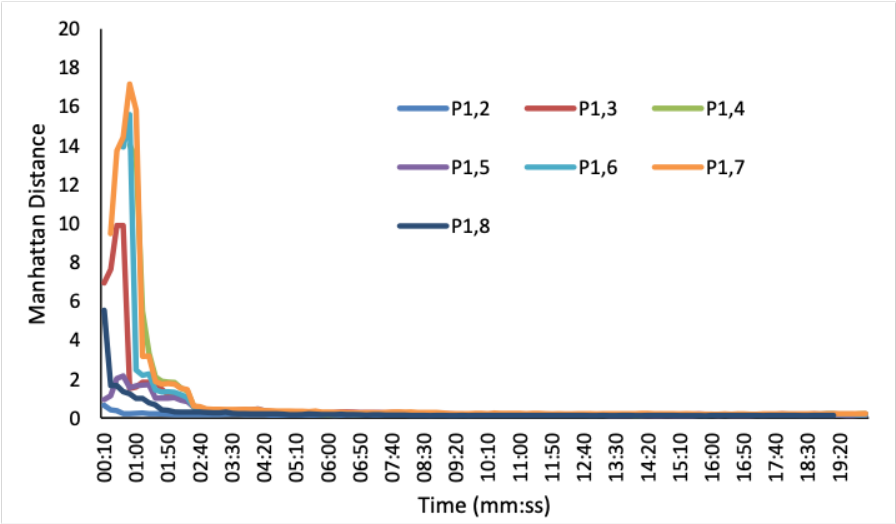


Figure 3.3: Set up with two groups interacting within the same room in close proximity (Experiment 2). Left: an overhead view of the table setup where P_n is the seating position of a participant and p_n^* is a participant's device which was in the participant's pocket

Our aim was to target "challenging" scenarios where groups of people interact within close proximity of each other. We set up two experiments: "Experiment 1" representing a typical social interaction of a single group during a meeting, and "Experiment 2" where two groups were interacting within the same room in close proximity. Specifically, the first experiment involved 10 participants joining a meeting and sitting around a large table. The participants were asked to keep their phones on the table during the meeting (see Figure 3.2). The data collection app was installed on the participants' phones before joining the meeting. In the second experiment, two groups of participants where asked to engage in two separate meetings within the same room. The experiment involved 6 participants (3 female, 3 male) who were asked to join the meetings. The participants were split into two groups of 3 people each, sitting on two tables with no more than 1 meter distance between them (see Figure 3.3). The participants were asked to place their smartphones on the table during the meeting, and two participants (one from each group) were asked to place an additional smartphone in their pocket. The two groups were asked to engage in conversations while sitting in close proximity to the other group.

3.4.2 WiFi Signals



Above: Data of co-location during the meeting scenario, showing pattern of when users became co-located. If we zoom in (**below**), we can see refined distance metrics between devices

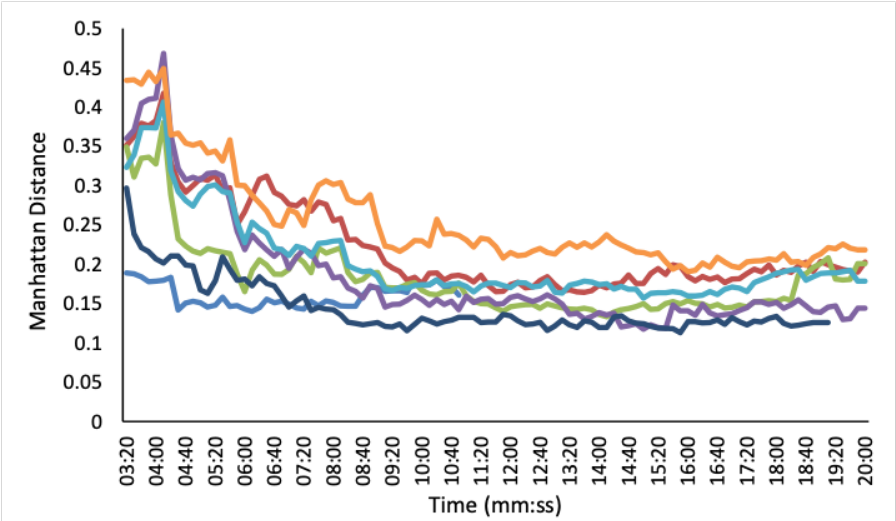


Figure 3.4: In this graph, each line refers to Participant 1 (P1) and the Manhattan distance between a corresponding participant. For example, for the distance between P1 and P2, this has been labelled as P1,2. The positions of each participant are outlined in figure 3.2
Top: Data of co-location during the meeting scenario, showing the pattern of co-location.
Bottom: Data from the same meeting, cropped from the section where all participants are interacting (minutes 3 to 20), and zoomed in

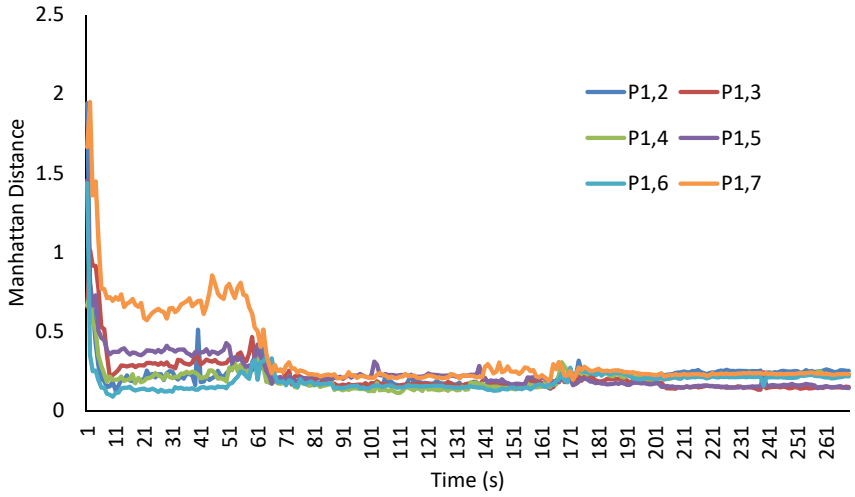


Figure 3.5: Graph showing the Manhattan distance over time for the closely co-located social groups. In this graph, each line refers to Participant 1 (P1) and the Manhattan distance between a corresponding participant. For example, for the distance between P1 and P2, this has been labelled as $P1, 2$

. The groups for this social interaction are outlined in figure 3.3.

Following similar work from [107], we explored how WiFi fingerprinting can be utilised to detect co-location in these experiments. Specifically, every smartphone device scans periodically every 10s for available WiFi access points which transmits at 2.4 GHz, and collects the received signal strength indicator (RSSI) and basic service set identifier (BSSID) for each access point response. These values are then used to generate a WiFi fingerprint for each participants' phone and subsequently estimate co-location between participants.

3.4.2.1 WiFi Proximity

A WiFi scan performed at time t generates a signal strength vector

$$S_t = \{ap_1 : rss_1, \ldots, ap_n : rss_n\}$$

where each access point ap is identified by its MAC address, rss is the received signal strength value for ap. We generate a WiFi fingerprint for the smartphone of each participant, by aggregating multiple WiFi scans over a sliding window of $w = 60s$

with 33% overlap. Consider $SW_t = \{S_i : i \in (t-w, t)\}$ to be the set of subsequent scans within the window w. The WiFi fingerprint at time t is:

$$F_t = \{ap_1 : rss'_1, \ldots, ap_n : rss'_n\},$$

where $ap_i \in SW_t, \forall ap_i, rss'_i = avg(rss_i : rss_i \in SW_t)$. As it was shown in [109], the RSSI values captured by different smartphone models can vary significantly even when collected under identical conditions. In order to allow appropriate comparisons between WiFi fingerprints, the recorded RSSI values are normalised and converted to positive scale, by dividing them with the maximum RSSI within the fingerprint:

$$rss_i^{norm} = \frac{rss'_i + 100}{rss'_{max} + 100}$$

where rss'_i is the RSSI value for access point i i and rss'_{max} is the maximum RSSI value for the entries of averaged fingerprints.

The generated fingerprints are then used to estimate the proximity between participants. Specifically, fingerprints generated by different participants are compared using a similarity function. We assume that the level of similarity is an indication of proximity between these participants. We applied the Manhattan distance as a similarity function, as it demonstrated the highest level of discrimination between different co-location distances when compared to Euclidean distance and Tanimoto similarity. Specifically, for any two fingerprints that were compared, each fingerprint was extended by adding access points that only appeared in the other fingerprint. The added access points were given an RSS value of 0dB. The similarity metric between the fingerprints was given by the Manhattan distance, with an additional division of common count to provide scaling [107]:

$$distance = \frac{1}{n} \sum_{i=1}^{n} |rss_i^a - rss_i^b|$$

where n is the number of elements in the intersection between the two fingerprint sets: $n = |A \cap B|$, and rss_i^a and rss_i^b are the normalised RSS values for the access point i captured by the two devices a and b.

We used the WiFi scanning dataset to estimate the distance metric of the participants in the two controlled experiments. In Experiment 1, all participants joined a group meeting in the same room. The pair-wise distance metric for all participants over time clearly shows that the WiFi fingerprint can identify the participants joining the meeting (Figure 3.4). Indeed the WiFi fingerprint comparison can clearly capture the sequence of arrival of the participants for example. However, when exploring the WiFi fingerprint similarities during the meeting we can see significant variations although the participants did not change their location during the meeting (Figure 3.4).

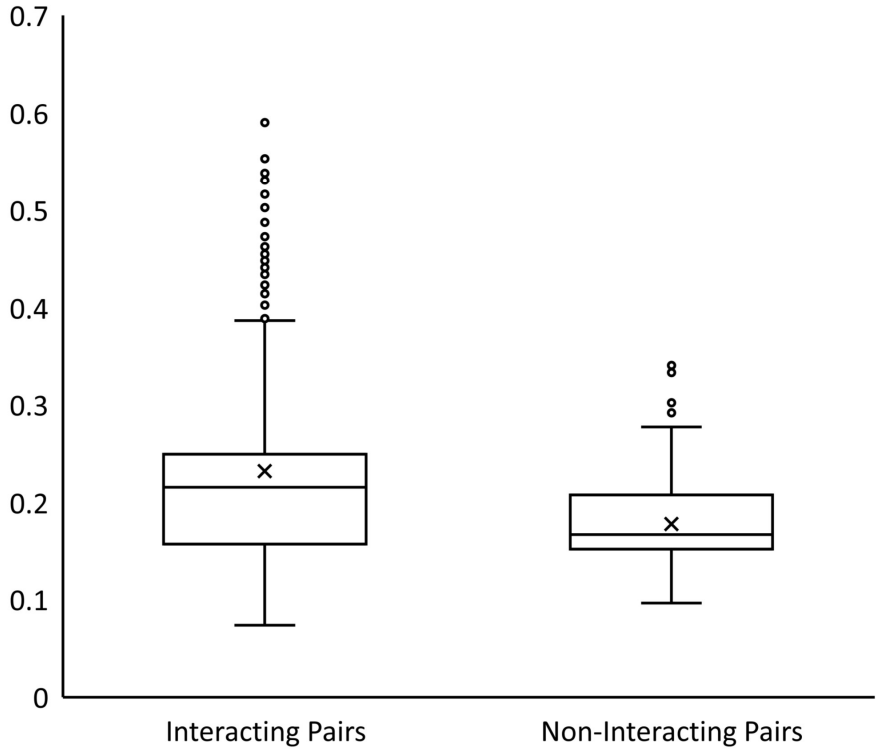


Figure 3.6: Box diagram showing the mean and spread of similarity values for interacting pairs vs non-interacting pairs. The similarity value on the y-axis is Tanimoto Similarity

In our second experiment we attempt to explore how WiFi fingerprinting could be applied in the case of co-located social interactions. In that experiment, we have two groups of 3 people each, interacting in close proximity to each other (less 1 meter distance between the groups). Looking at the similarity of the WiFi fingerprints (Figure 3.5), there is no obvious pattern that helps identify the two interacting groups. Furthermore, we explored the overall distribution of the similarity measurement

(as Manhattan distance) between the pairs of participants that were interacting with each other, and compared it with the distribution of measurements between pairs that were not interacting. As shown in Figure 3.6, the two distributions may have slightly different median value, however, both distributions overlap significantly. This is a clear indication that for such close proximity between social groups, WiFi fingerprinting alone may not be sufficient to distinguish the two groups.

3.4.3 Audio Signals

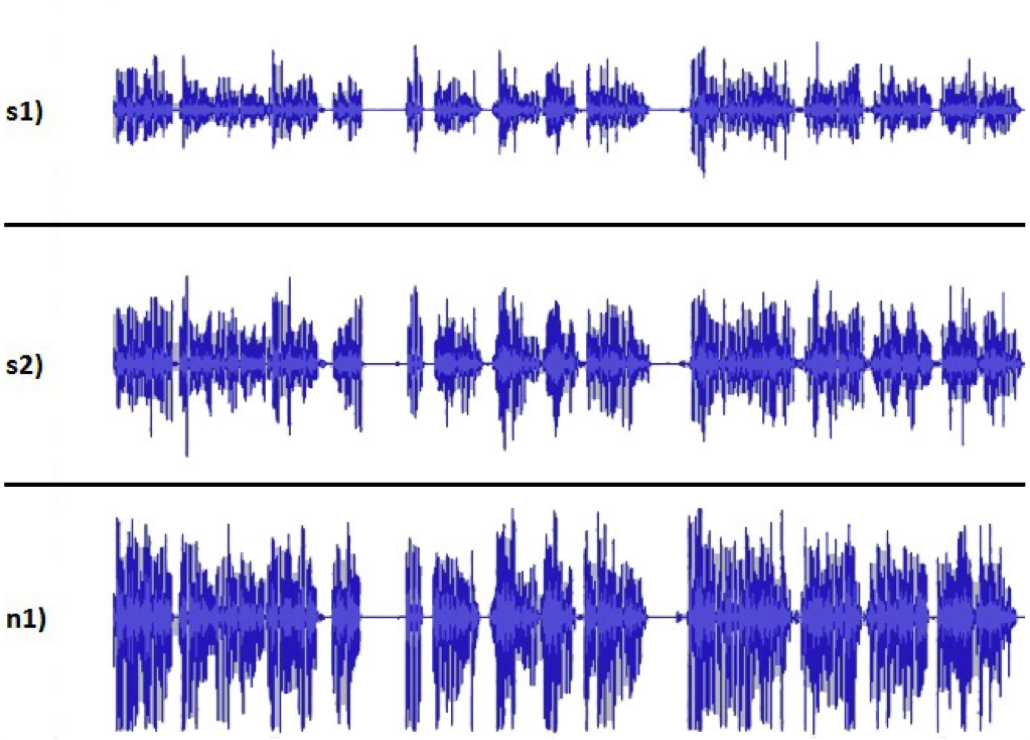
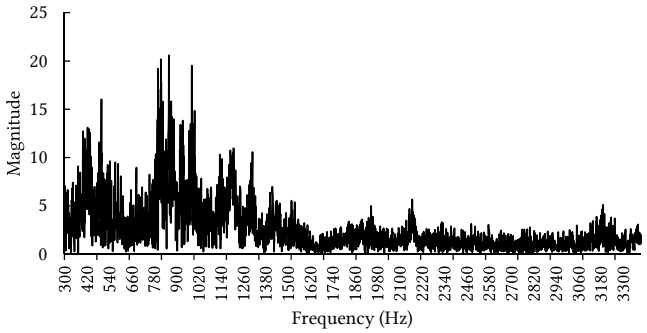
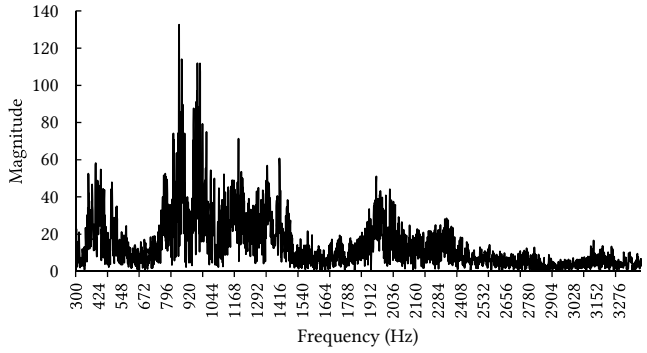


Figure 3.7: Sound waveforms captured by two Samsung S5 devices "s1", "s2" and a Nexus 5 "n1" recording the same speech segments at the same distance from the source. The devices have different gains.

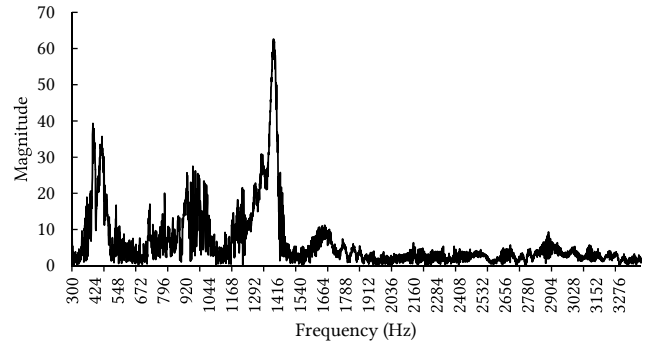
Considering the limitations of using WiFi signals alone to detect social interactions, we explored the feasibility of relying on the capture of audio signals as a way of distinguishing social groups that are in close proximity to each other. One of our early intuitions was to explore how the amplitude of the sound signals can be used to identify the distance of the device from a speaker. This was an approach similar to the work shown in [114]. Before attempting to analyse audio sensing during the social interaction experiments, we first explored how audio signals are captured by



(a) A sample of frequency domain data from a device placed on “table a”



(b) A sample of frequency domain data from a different device placed on “table a”



(c) A sample of frequency domain data for a device placed on “table b”

Figure 3.8: Frequency spectrums of audio samples from different devices in Experiment 2, captured during the same time window.

different devices. We set up a test where a range of smartphone models were placed at the same distance from a speaker (approximately 1m) and captured the same audio of a human speaking. The smartphone devices used included two Samsung S5, two Nexus 5, a Motorola G and a Vodafone Smart Ultra 7. In order to avoid any irregular shaping of the captured signal, we ensured that the auto-gain function on the devices was not active. Audio was captured at 16 KHz sampling rate. We manually extracted the speech segments from the audio recordings and inspected the captured audio signals. As shown in Figure 3.7 different devices capture sound signals at different energy levels. Even the same models, can have significant differences. For example, the two Samsung S5 devices captured the same speech sounds at an average amplitude of -28.59 dB and -38.53 dB, while the Nexus 5 captured the same source at -34.24 dB. Such difference could be attributed to differences in the hardware design, wear and tear, or just dust that is accumulated around the microphone. These results demonstrate that to utilise the volume of the captured sound to estimate distance would require extensive calibration to identify a normalisation coefficient for each device. Although some solutions exist to calibrate the different gains between the microphones [115], this would require a supervised calibration phase whenever new smartphones are introduced to the system, leading to a system that is not scalable and not fit for unsupervised deployment. Furthermore, the differences between same phone models show that it would not be possible to construct a generic calibration database for different phone models.

As amplitude alone was not considered a sufficient feature to identify social groups, we attempted to explore if sound signals can reveal distinctive patterns that can help differentiate between people participating in the same conversation. In Experiment 2, participants were asked to place their phones on the table in front of them (tables a and b), while 2 participants had phones placed in their pockets (Figure 3.3). We analysed the sound signals captured by these devices, extracted samples of audio of 2 seconds and applied a Fast Fourier transformation [116] (FFT) to look at the sound patterns in the frequency domain. We selected the frequencies between 300 Hz and 3,400 Hz, which is considered the speech range [117]. This filtering allowed us to eliminate some of the noisy data that was captured by the phones

in participants' pockets. Figure 3.8 shows the frequency patterns from two devices participating in the same conversation (Figures 3.8a and 3.8b), and a device on a different conversation nearby (Figure 3.8c). The patterns that we observe in this case show that devices participating in the same conversation have high energies around the same frequencies, while devices on different conversations show a significantly different pattern. Based on these observations, it is possible to devise a technique to extract a *"sound fingerprint"* that is based on the most significant frequencies of captured audio data, that can help distinguish between users participating in different conversations. Although the experiment involved participants in very close location, the difference in sound patterns can be explained by the natural tendency of people to speak loud enough so that all of their interacting participants can hear them, also known as the "Lombard effect" [118]. This in practice ensures that the sound captured by the phones in close proximity to the conversation is dominated by the speakers participating in that particular group.

Based on the findings of these preliminary studies, we aimed to design a system to detect social interactions using a combination of WiFi signals, as an early indicator that users are in close proximity, followed by audio sensing to identify smaller groups within the same area.

3.5 System Overview

Next2Me is a mobile sensing system that can identify social interactions using WiFi and audio signals. The overall architecture can be seen in Figure 3.1. The system consists of sensing components running on a smartphone device, and a cloud service responsible for comparing datasets from multiple users. The system relies on WiFi fingerprinting to discover when a user is co-located with other users of the system. When co-location is detected, the participant's smartphones are triggered to perform sound sensing. The sound sensing subsystem is responsible for discovering similarities in the sound patterns captured by the participating smartphones. The sound similarities are then used to identify a social network, as it is formed by the similarities of the sound signals. Finally, applying a community detection algorithm

helps identify the sub-groups of people interacting within close proximity to each other. The following sections describe the system in more detail.

3.5.1 WiFi and Co-location

The system relies on WiFi fingerprinting to detect co-location between users. Each smartphone device scans every 10s for nearby WiFi access points transmitting at 2.4 GHz. Using the signals strength information gathered from nearby access points, we construct a WiFi fingerprint as it was described in Section 3.4.2. The aggregated WiFi fingerprints, containing the normalised average signal strength values of access points over a window of 60s, are uploaded to the cloud. A cloud service is then responsible for estimating if two devices are co-located. Specifically, an adjusted Manhattan distance metric (as shown in Section 3.4.2) is calculated over the WiFi fingerprints of smartphones that are potentially co-located (i.e. contain at least one common access point in their set). Subsequent WiFi fingerprints are generated every 2.5 mins.

Deciding if two users are potentially participating in a social interaction according to proximity depends on two parameters: the estimated distance between them, and the duration that they are co-located for. The selection of these parameters depends on the particular types of interactions that are targeted by the system. In this system, our objective is to capture significant social interactions that last for more than a few minutes. Specifically, we consider two devices to be co-located if the Manhattan distance is below a threshold of 0.8. Based on our preliminary experiments, the threshold was considered sufficient to discover co-location within less than 5m. Furthermore, if two users are co-located for more than a period of 5 mins, we consider this a potentially significant interaction. If these conditions are met, the cloud service triggers the co-located phones to initiate their sound sensing tasks.

3.5.2 Sound Fingerprint

The preliminary analysis of audio signals showed that smartphones that are close to a social interaction can capture distinctive frequency patterns that can help distinguish the nearby social groups. In order to capture such patterns, we designed

a technique that can capture a *"sound fingerprint"* that can represent the speech patterns detected over a time window of a few seconds. Our aim was to represent the sequence of sounds over that window as a fingerprint vector that can be easily compared with other fingerprints captured by nearby smartphones.

The sound sensing subsystem of the Next2Me system captures audio at a sampling rate of 16 KHz. This allows a Fast Fourier Transform (FFT) resolution of 8 kHz, but also provides a balance of higher quality signals. We use a window size of 2 secs, with a hamming window for calculating the FFT. The window of 2 secs was considered sufficiently large to allow more lenience with audio synchronisation across smartphones, considering that the on-board real-time clocks may not be perfectly synchronised. We extract the frequency bands between 300 Hz and 3,400 Hz which is the typical spectrum for human speech.

Considering the results from the preliminary study, we can observe that frequency spectrums from devices around the same social interactions demonstrate high magnitudes around the same frequencies. Our aim is to use the significant frequencies in each sound sample as a way of comparing the sound patterns captured by different devices. As the sound capturing sensitivity varies across devices, we first need to reduce the variance on the sound spectrum that is produced. We apply a linearly weighted sliding average across the spectrum to smooth the results. Next, we subsample the smoothed spectrum to reduce the granularity. Specifically, we use a 30 Hz spectral window and calculate the average frequency magnitude for each window. The whole process produces a smoother frequency spectrum with 30 Hz granularity. From this spectrum, we define as *partial fingerprint* the set of the top n frequencies with the highest magnitude.

In order to improve the robustness of the fingerprint against ambient noise, we produce the *sound fingerprint* for part of a social interaction by combining multiple partial fingerprints as a time series of sets with the top n frequencies (see Figure 3.9):

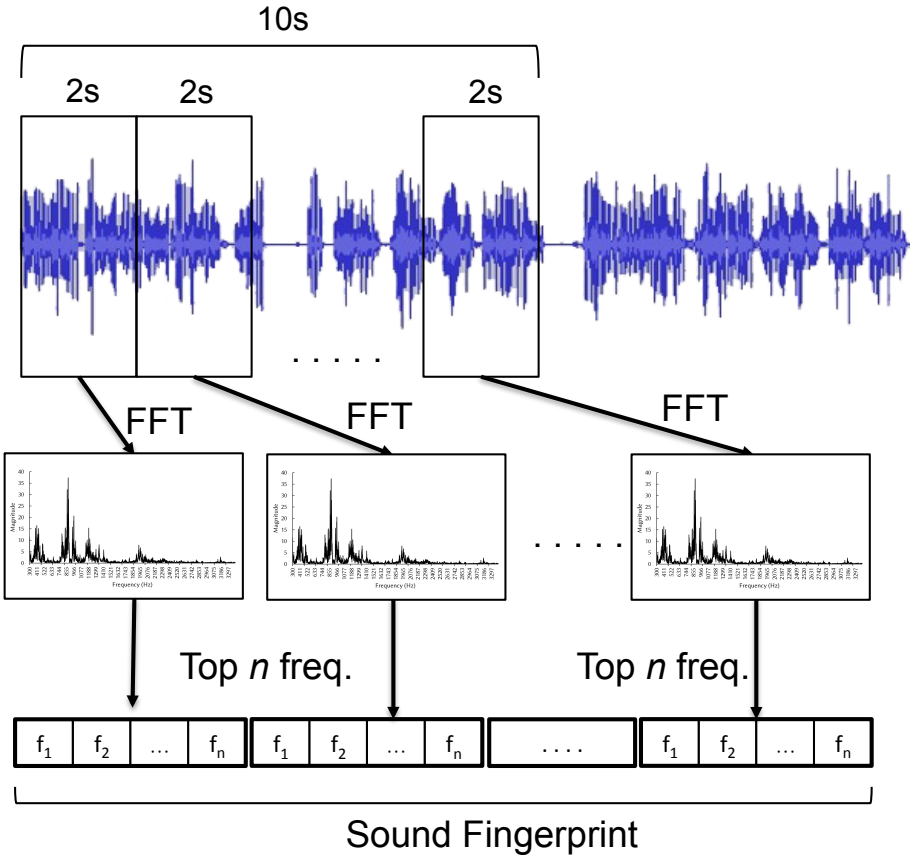


Figure 3.9: Complete overview of the process of generating a sound fingerprint that represents 10s of captured audio. In this, the audio sample is recorded, sliced into subsampled, converted to an FFT (for frequency spectrum) and then the top n frequencies are extracted as a sound fingerprint

$$S = \{P_1, P_2, \ldots, P_k\}, \text{ where } P_i = \{f_1, f_2, \ldots, f_n\}$$

devices can be compared by using the Jaccard index over their partial fingerprints. The Jaccard index ($\frac{|A \cap B|}{|A \cup B|}$) measures the similarity of two sets by estimating the number of common frequencies over the total number of unique frequencies in the two sets. We define the similarity function for two sound fingerprints as the average Jaccard index of their partial fingerprints:

$$sim(S_a, S_b) = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{|P_a^i \cap P_b^i|}{|P_a^i \cup P_b^i|} \right)$$

The output of the comparison of sound fingerprints gives us a metric that represents the proximity of people according to the sounds captured by their phones.

3.5.3 Community Detection

The sound fingerprints captured by the smartphones are uploaded to the cloud. A cloud service calculates and estimates the similarity metric between sound fingerprints of all the co-located devices. This similarity metric is then used to produce a weighted graph that represents a social network of all the co-located devices. It is expected that smartphones of users participating in the same social group will have a higher similarity (i.e. weight) in the graph. Using the social graph, we attempt to extract the separate groups as "communities" by applying the Louvain community detection algorithm [119].

The Louvain community detection algorithm is a method for detecting communities within networks. It assigns a modularity score for each community, where the modularity measures the quality of a node's assignment to communities, and places nodes within their best matching community based on this by evaluating how dense of the connected nodes within a community are. This acts as a hierarchical clustering algorithm, which recursively assigns nodes into clusters on a graph.

Fortunato et al. [120] showed that the modularity used in the Louvain community detection algorithm suffers from a resolution limit, which means that by optimising the modularity, communities that are smaller than a specific scale cannot be resolved. In order to overcome the resolution limit issue experienced, we used the resolution limit technique described in [121]. We experimentally chose a limit of 0.8 to allow smaller communities to be identified. The output of the community detection represents the output of the system, identifying the different groups interacting within close proximity of each other (Figure 3.10).

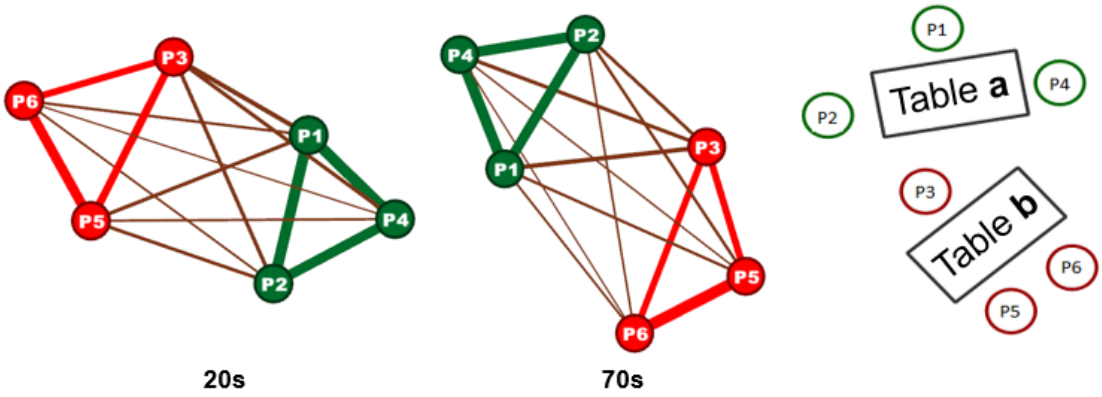


Figure 3.10: Community network graphs over time, at different time points since the start of the interaction (Experiment 2). Node colours reflect the grouping produced by the community detection algorithm. P_n refers to the participant number. To the right, an illustration of tables shows the seating positions of participants

These social graphs illustrate the relationships between social groups who are interacting, over time. We plot the strength of the links between people using lines, which are thicker according to the weight between them. Using this, we can then apply a community algorithm to separate the groups based on the weight between connected nodes, to visualise the different social groups as a collection of nodes. In Figure 3.10, this is shown using red and green nodes. We also show what happens to these groups over time, at 20 seconds and then 90 seconds.

3.5.4 Fine-tuning parameters

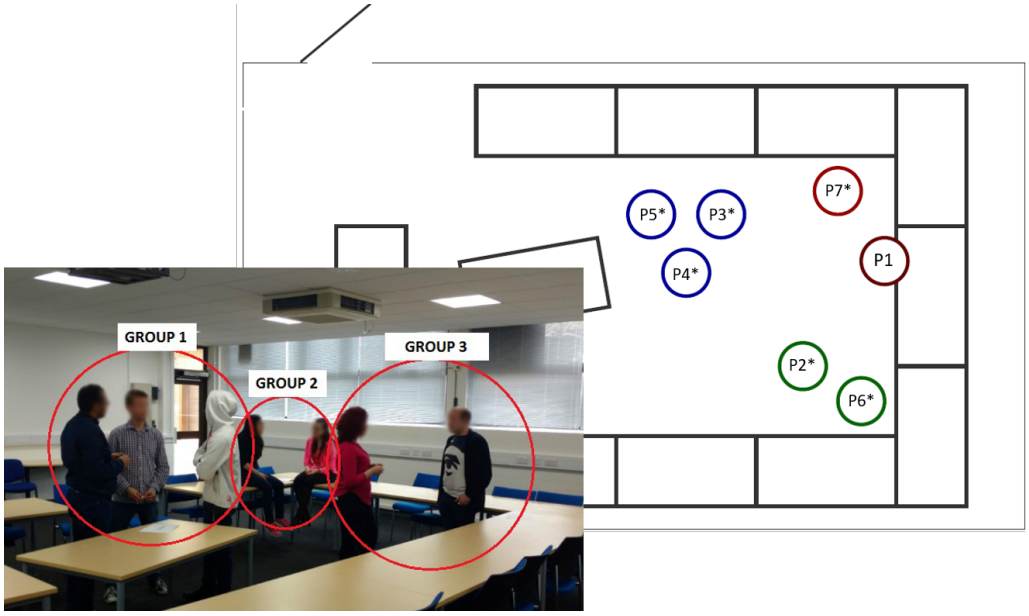


Figure 3.11: Example layout for the conference scenario during Stage 1. The red circle separates different social groups. Right: an overhead view of the participants layout during this experiment.

Stage	**Groups**		
1	(P1, P2, P3)	(P4, P5)	(P6, P7)
2	(P1, P2, P6, P7)	(P3, P4, P5)	–
3	(P2, P6)	(P1, P7)	(P3, P4, P5)

Table 3.2: The groupings of participants during the networking experiment. Participants changed the formed groups three times during the experiment. All participants had their phones in their pockets.

In order to analyse and fine-tune the parameters of the system, we needed a scenario that involves a more complex setting than the preliminary studies. We conducted an additional study involving a social networking event (Experiment 3). We invited 7 participants (2 female, 5 male) to join a large meeting room and engage in a typical networking situation where they were asked to form smaller groups and freely discuss about their work (Figure 3.11). The participants had the Next2Me app installed on their smartphone devices. All participants kept their smartphones in their pockets during the event. At regular intervals participants were asked to "mingle", changing the groups of people they talk with. Throughout the event, an observer kept track of

the ground truth marking the actual groups that were formed. During the scenario, the groupings changed three times as shown in Table 3.2.

Using the dataset captured through this scenario, we attempted to fine-tune the parameters used for the sound fingerprint. Specifically, to identify the number of top frequencies selected for the partial fingerprints, and the total length of the sound fingerprint. In order to assess the quality of each configuration we calculated the number of nodes that were grouped with the majority of their correct social peers, against the number of nodes that were incorrectly placed in a group that did not involve their correct social peers. Throughout our analysis, we use the precision of the results: $\frac{C}{C+I}$ where C is the number of correctly grouped nodes, and I is the number of incorrectly grouped nodes. In these estimations, the notion of *false positive* and *false negative* are essentially the same.

In order to estimate the best parameters for the sound fingerprint, we selected a small sample of audio data captured in this experiment where there was definitely speech. Using a combination of numbers for the n top frequencies, and the total duration of the fingerprint, we achieved the best results when selecting the top 6 frequencies for each partial fingerprint and maintaining a fingerprint window of 10secs.

3.6 Evaluation

In a real scenario, the Next2Me detection of social interactions does not necessarily need to run continuously throughout a co-location event. Instead, audio sensing can be used sparsely during a period to identify social groups. In our evaluation, we firstly attempted to estimate the average precision that can be achieved if sound fingerprinting is used only once during a social interaction using a 10 secs fingerprint. As it is shown in Figure 3.10, applying the sound fingerprinting technique at different intervals can have varying results. To estimate the average performance of the system, for each experiment we calculated the average performance for every 10 secs time window of the social interaction, using a 10 secs sliding window with 9 secs overlap.

For Experiment 2, two tables were positioned no more than 1m apart, phones were placed on the table, and two participants had additional phones in their pockets.

Experiment	Precision	Correct	Incorrect
Exp 2 - On table	1.00	66	0
Exp 2 - In pocket	0.88	99	13
Exp 3 - Stage 1	0.74	149	50
Exp 3 - Stage 2	0.91	238	21
Exp 3 - Stage 3	0.80	186	44

Table 3.3: Results for the two experiments involving social interactions of groups in close proximity. The precision is calculated from the number of correct and incorrect samples being classified by the system

We first estimated the average precision by including only the smartphones that were placed the desks, which resulted in 100% success rate (Table 3.3). This is a good result but somehow expected, considering that the experiment was performed in a quiet environment, and the smartphones were in the centre of conversations that were taking place. When combining the system with the smartphones placed in pockets, the average precision dropped to 88.3%. Exploring the results, we could see that the location of one pocket smartphones was quite close the second group, occasionally picking up stronger sound signals from the other table. Furthermore, the table itself acted as a barrier, blocking sound signals from the conversations reaching the pocket smartphones affecting the precision of the overall system.

In Experiment 3, where participants socialised within the same room, all smartphones were placed in participants' pockets. We ran our evaluation over the different stages where different groups were formed. The overall precision ranged from 74% to 91% (Table 3.3). Although these were encouraging results, they all relied on capturing a single sound fingerprint during a social interaction. Next, we explored how combining multiple sound fingerprints could improve the overall precision of the system.

3.6.1 Duty Cycling

We anticipate that precision on social interaction detection using sound fingerprints can be improved by combining multiple sound fingerprints captured over longer periods of time. There are cases where a randomly selected 10sec sound fingerprint may capture a situation where the actual social groups are not correctly mapped. Allowing more than one sound fingerprints to be inspected at different time points, offers more chances to discover the correct social group mapping. By applying a

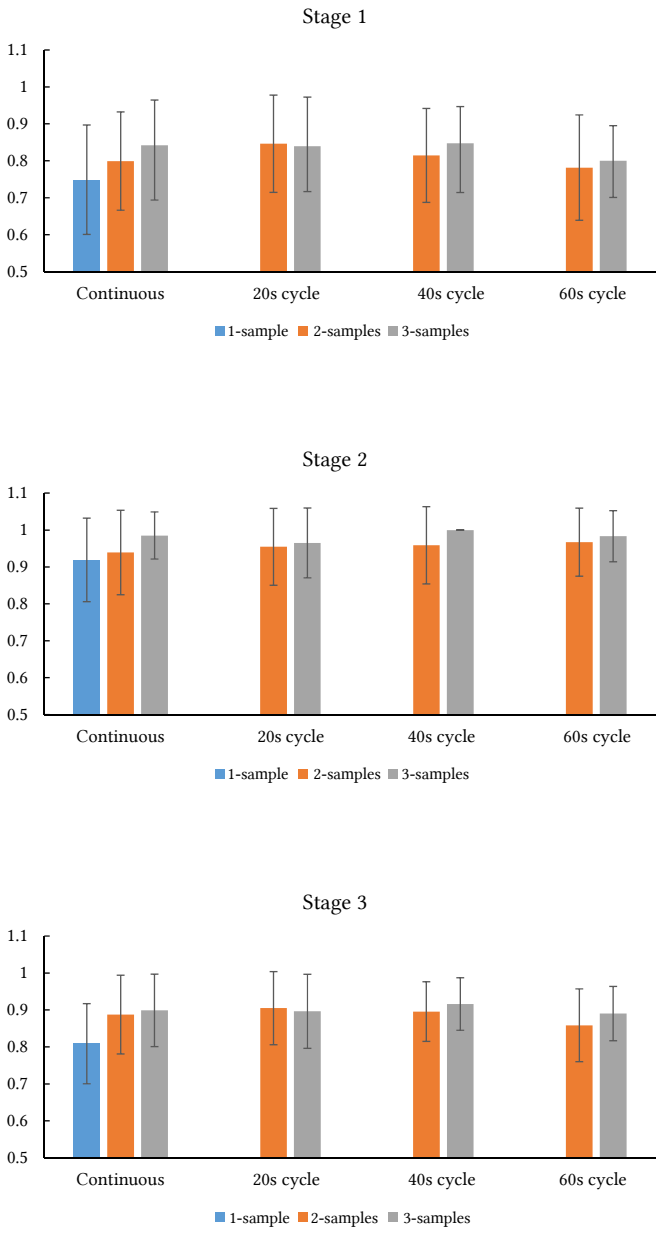


Figure 3.12: Effect of duty-cycling window size and number of sound sensing samples in the overall precision.

duty cycling scheme, there are ways to potentially improve the overall precision of the system while keeping the energy cost relatively low. Specifically, we explored the effects of a fixed-length duty cycled sensing, where a fixed number of sound fingerprints can be captured during a potential social interaction. When combining multiple fingerprints, we wanted to explore what is the number of consecutive sound fingerprints that we should use to improve precision, and how the length of the sleeping windows between them would affect the overall result.

Combining multiple fingerprints for the detection of social groups would involve modifications in the way that the weights in the social network graph are calculated. Specifically, when the social graphs are formed, the weight between two nodes includes the average fingerprint similarity over the number of sound fingerprints:

$$W_{a,b} = \frac{1}{k} \sum_{i=1}^{k} sim(S_{i,a}, S_{i,b})$$

where, $W_{a,b}$ is the weight between participants a and b, k is the number of sound fingerprints involved, and $S_{i,a}$ is the i-th sound fingerprint for participant a. After weights are estimated by combining multiple fingerprints, the same community detection algorithm is used to estimate the social groups that are formed.

Using the datasets from Experiment 3 (networking event), we tested the performance of the system when using 2 or 3 sound fingerprints, with a varying sleeping windows between them; ranging from continuous (no sleeping) to fingerprints captured with a 60sec gap between them. We calculated the performance of the system, with the duty-cycled scheme being applied at any time during the whole experiment and estimated the average precision of the results (Figure 3.12). The results show that using more than one sound fingerprint improves the overall precision, while the combination of three fingerprints reduces the variance that we observed in precision. Generally, we see that combining multiple fingerprints with a sleeping window of 40sec offers the best results. Specifically, a duty cycling scheme of 3 sound fingerprints with 40sec sleeping shows an average precision of 92% and a combination of 2 sound fingerprints with 40sec shows an average precision of 89%. Following this, we conclude that for a setup of 3 samples/40sec sleeping is appropriate for the high precision, while a 2

samples/40sec sleeping scheme offers a good balance of energy cost and precision.

3.6.2 Coffee Shop scenario

Figure 3.13: Experiment setup for the Café social interaction. Left: an overhead layout of the experimental setup, where grey nodes indicate other people who were in the same social space and the green and blue nodes represent the two social groups who participated in the experiment

As a final step in the evaluation of the Next2Me system, we performed a *"real-world"* deployment where participants were involved in social interaction within a busy coffee shop. Six participants were invited to install the Next2Me app onto their phones and to meet in a busy coffee shop and socialise, forming two separate social groups and sitting at nearby tables (Figure 3.13). The event took place during a busy time where a number of other people were in the coffee shop, and engaged in conversations. The setup was selected to ensure the environment involved ambient noise of other people talking to each other. During the event, participants placed their phones on the table while two participants had a smartphone placed in their pocket. Note that the table in this scenario had a relatively lower height than the tables involved in previous scenarios.

The system was configured to perform WiFi scans to detect co-location, and trigger

sound fingerprinting when participants were co-located for more than 5mins. The overall experiment lasted for 20 mins. We analysed the performance of the system using 3 sound fingerprints captured with a 40sec sleeping window between them. Using the devices involved in the scenario, the system achieves an average precision of 88%. When the pocket phones are not included in the estimation, the precision raises to 99.1%. This shows that phones situated without physical obstructions and in an open environment will perform well, and smartphones in a pocket will be clustered into communities with less precision due to the frequency-filtering effect of the pocket material.

3.7 Energy Considerations

The design of the Next2Me system relies heavily on sensing modalities that can have a significant impact on the battery life of the participants' smartphones. In this section we analyse the energy cost implications of using Next2Me. In our analysis we attempt to establish the average cost in the form of electric charge (measured in mAh) consumed by the Next2Me during a day. This estimation will allow us to consider the impact that the system would have on the battery life of common smartphones, with battery capacities in the rage of 2,800mAh (Samsung S5) to 3,220mAh (Nexus 6).

The WiFi fingerprinting subsystem relies on the periodic WiFi scanning to discover near-by WiFi access points. If we consider that the electric charge consumed during a WiFi scan is E_w, a WiFi fingerprint is generated using 6 scans, and a fingerprint is produced every s_w seconds, the overall cost of continuously running the WiFi scanning subsystem for a whole day is:

$$W_{total} = \frac{86,400}{s_w}(6 \cdot E_w + N_w) \tag{3.1}$$

where 86,400 is the total number of seconds in a day and N_w is the average energy cost of uploading data to the cloud.

When a co-location incident is captured by WiFi scanning, and it lasts for more than

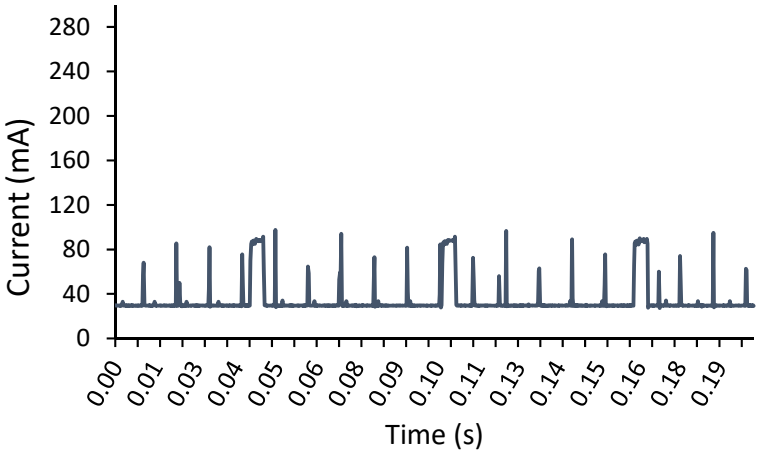


Figure 3.14: Current over time for the continuous recording of audio at 16kHz sampling rate

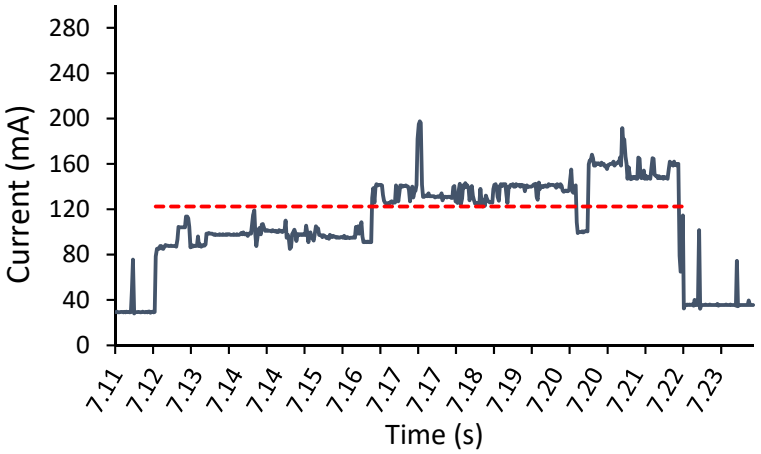


Figure 3.15: Current over time for one FFT of a 2-second audio sample (includes audio sensing and baseline).

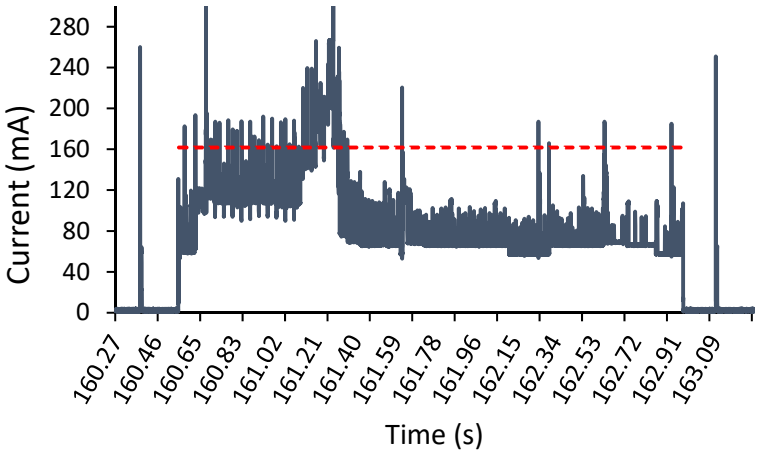


Figure 3.16: Current consumption over time for one Wi-Fi scan

5mins, the sound fingerprinting subsystem is triggered. Each sound fingerprinting will involve 10sec of audio recording, and involves a CPU processing cost to perform a FFT over the sample. Subsequent sound fingerprints will then be uploaded to the cloud for comparison. We can model the additional energy for the sound fingerprinting subsystem caused by a single social interaction as:

$$S_{int} = 3 \cdot (E_{sense} + E_{FFT}) + N_s \tag{3.2}$$

where S_{int} is the cost for a single social interaction, E_{sense} and E_{FFT} are the costs for capturing audio for a sound fingerprint, and performing the FFT respectively. The data from 3 sound fingerprints would typically be uploaded to the cloud, incurring an additional N_s cost for network communication. From Equations (3.1) and (3.2) we can estimate the average energy for an individual who has on average k significant social interactions during their day:

$$E_{total} = W_{total} + k \cdot S_{int} \tag{3.3}$$

In order to estimate the average energy of the Next2Me system, we performed a number of lab measurements to estimate the energy consumption. We used the Monsoon Power Meter setup to intercept the current drawn from the battery of a phone. We run experiments using the Samsung Galaxy J3 smartphone. A base line current when a phone is not performing any activities was estimated to be 9.16mA (2.27mA in airplane mode). When the phone was set to perform WiFi scanning, the average current during the scanning, without the baseline, was estimated to be 93.24mA. Each scan lasted for approximately 0.78s which results in an electric charge of $E_w = 72.73$mAs. The average cost of data upload can vary significantly depending on the network infrastructure and external conditions. In order to estimate the impact of data upload using WiFi we use the energy cost per KB of 5mJ as it is estimated in [122] which results in consumed energy charge of $N_w = 1.3$mAs. In the final deployment of the system we set the WiFi scanning subsystem to perform a scan once very 2.5 mins (which would enable the detection of 5min colocation instances). From equation 3.1 we can then estimate that in case of a user who does

not have any significant interactions during the day, the overall energy cost is:

$$W_{total} = \frac{86,400}{150}(436.38 + 1.3) = 252,138\text{mAs} = 70.03\text{mAh}$$

For a phone with a battery of 2,800mAh this would be 2.5% of the battery's capacity.

In order to estimate the impact of sound fingerprinting, we calculated the average energy cost of audio sampling, and performing a FFT over a sound sample. The average current for audio sampling without the baseline was estimated to be 32.84mA. For capturing an audio sample of 10s this would result in consuming an electric charge of $E_sense = 328.41mAs$. When performing a FFT over a 2s audio sample the average current (excluding the cost of audio sensing and baseline) is 56.14mA and the duration is 105ms. Therefore the cost of performing 5 FFTs for a sound sample of 10s would be $E_{FFT} = 29.47mAs$. From equation 3.2 we can estimate the additional cost of detecting a single social interaction as:

$$S_{int} = 3 \cdot (328.41 + 29.47) + 1.3 = 1074.94\text{mAs} = 0.29\text{mAh}$$

Assuming a case of a user who has about 20 significant interactions during the day, the additional energy capacity consumed by the system would be $E_{total} = 70.03 + 20 \cdot 0.29 = 75.83$mAh. This results to 2.7% of the battery's capacity. These results demonstrate that Next2Me has a very small impact on the smartphone's battery life and would be appropriate for continuous sensing.

3.8 Discussion

In the design of Next2Me we focus on the detection of significant social interactions, that last for more than 5-mins. The technique is robust for such social events, but would not be appropriate for capturing short time, serendipitous interactions that last for only a few seconds. Although such short interactions are beyond the scope of this work, the proposed technique could potentially be adapted with more aggressive use of sound sensing to capture such short events. However, such approach would increase the power cost of sound sensing, and would require further exploration in

adaptive sensing approaches to mitigate energy issues. Furthermore, the physical environment can have a significant impact on the performance of Next2Me. In this work we demonstrate that the proposed system is robust against smartphone placement in participants' pockets, but further investigation would be necessary to fully explore the impact of the environment, such as higher/lower ceilings or significant acoustic echo.

3.8.1 Continuous Sensing System

When building a continuous and automated sensing system for Android devices, there are a few things to consider:

- How many hours can the system run without the device needing to recharge?
- Are the participants comfortable with an intrusive sensing system?
- Are there any technological constraints to plausibility of running a continuous sensing system?

In order to better understand the limitations of implementing a continuous WiFi-based sensing system on Android, we developed and deployed an app entitled *SocialSense* to a group of participants from Universiti Teknologi MARA in Malaysia. *SocialSense* was an Android Application that was developed to track the human social interactions between app users who install the application. We recruited participants from the Universiti Teknologi MARA. Participants were invited the join the study via an email to the faculty. Ethics approval was applied for at the University of Kent Faculties Support Office, and was approved (ref: 0151617).

We deployed the app to 22 participants using any Android device version 4.2+ (API 17+). The RAM usage of the app was calculated as 30 MB on average, and the install size was calculated 10.84 MB. Other than this, other important factors were calculated as such:

- 1 WiFi fingerprint sample every 10 seconds (120,960 entries per user, per study duration)
- 1 data upload every 10 minutes (2016 entries per user, per study duration)

- 1 GPS entry every 30 minutes (672 entries per user, per study duration)
- Power consumption calculated as 24 mAh per hour
- Data upload size calculated as 164.16 KB per hour

This experiment was needed generally to understand the plausibility when deploying sensing system using Android devices, and to help understand what limitations exist during this deployment.

3.8.1.1 Sensor Accessibility Issues

We had 11 participants with consistent data points, but the 10 other participants were unable to collect any data throughout the duration of the study. We debugged the application to establish what had gone wrong.

Since Android Marshmallow, Google decided that in order to receive a scan result from WiFi, the device must also have GPS turned on. This is in addition to the user accepting the location permissions at run-time. If there is no GPS turned on, a scan result will return as an empty list. This behaviour also applies to Bluetooth sensors, and is most likely why so many devices didn't return any WiFi data for the entirety of the Malaysia study. The reason for these sensor accessibility changes, is because Google is trying to protect people from this kind of tracking - however, at the time of the user study, the changes were undocumented and no errors were in the output of the apps debugger. It was invisible, except for an issue which somebody raised in the Google Code website forum [123].

What can be learned here, is that the user should enable "Location" setting on their devices so that continuous sensing applications can capture sensor data from the WiFi. It is therefore advisory that when the application is launched, a check for "Location" setting is made and the user is prompted to turn it on in order to proceed. Whilst the app is in the background, when "Location" setting is disabled, a notification can be displayed within the phone which lets the user know that tracking has stopped due to the "Location" setting being disabled.

3.8.1.2 Real Time Clock Issues

Another issue which was detected, was that devices were collecting data at the exact same time but assigning timestamps which were different. For example, if two devices are in the same room and both detect a nearby Access Point at the same time, one devices would record the timestamp for this event n seconds before or after the other device. In some cases, n can be more than 120 seconds.

What was happening here, was a phenomenon called "clock drift". Most computing devices are equipped with a hardware oscillator assisted computer clock. The frequency of the hardware oscillator determines the rate at which the clock runs [124]. The clock of an Android device can become inaccurate because the frequency of the hardware oscillator varies across time. Time synchronisation is crucial in a distributed continuous system; if two devices are out of sync then the collected data cannot be compared accurately in real time.

The way that we solved this was to implement a "Network Time Protocol" (NTP) [125] service, hosted by our back-end. NTP is a protocol for synchronising clocks by returning a precise and central timestamp. This means that every time a request is made for the most recent time by the app, the server will return the precise timestamp and add on the duration of how long the networking is estimated to take. Such a method is utilised in a few works [126, 127].

In Android, modifying the device timestamp is not allowed. Therefore, a request is made to the NTP server every 10 minutes and an offset is produced between the device clock and the precise clock provided by the NTP server. This offset is used to modify the timestamp for WiFi scan results of captured Access Points. This means that when data is compared, it utilises a central timestamp-based synchronisation method to try and keep things more accurate.

3.9 Conclusion

We developed a system that can use WiFi and Audio signals captured by smartphone devices to detect social interaction. The proposed system detects the social

interactions between people in various environments by capturing their co-location using WiFi, and accurately distinguishing social encounters through the capture and analysis of *"sound fingerprints"*.

In this chapter, we began by running some preliminary experiments to explore the extent to which WiFi based proximity detection can enable the identification of social interactions that occur between groups of people. Furthermore, we explored how audio signals can be analysed to further assist in identifying these types of social interactions. We analysed the results of a meeting experiment and a group experiment, and determined that using WiFi to capture was not sufficient enough to isolate the separate social groups from the analysis; the two distributions from our results had slightly different median values and overlapped significantly. This was a clear indication that for such close proximity between social groups, WiFi fingerprinting alone was not be sufficient to distinguish the two groups, and we instead tried a multi-modal technique by also analysing audio signals.

Considering the limitations of using WiFi signals alone to detect social interactions, we explored the feasibility of relying on the capture of audio signals as a way of distinguishing social groups that are in close proximity to each other. We explored if sound signals can reveal distinctive patterns that can help differentiate between people participating in the same conversation, and developed a technique of using the top n frequencies in a speech sample to generate *"sound fingerprints"*. Based on the findings of these preliminary studies, we designed a system which would detect social interactions using a combination of WiFi signals, as an early indicator that users are in close proximity, followed by audio sensing to identify smaller groups within the same area. We then fed the output of our social sensing system into a community detection algorithm to extract communities from he data by applying the Louvain community detection algorithm. The output of the community detection represents the output of the system, identifying the different groups interacting within close proximity of each other.

To improve the performance of our system whilst reducing the power consumption, we optimised the parameters for the system. We applied a duty cycling scheme.

Specifically, we explored the effects of a fixed-length duty cycled sensing, where a fixed number of sound fingerprints can be captured during a potential social interaction. The results showed that when using more than one *"sound fingerprints"* improved the overall precision, whilst the combination of three fingerprints reduced the variance that we observed in precision.

Finally, we tested our system in a real-world scenario to evaluate the performance of the system. This showed that our technique can achieve a high precision at low energy overhead, regardless of sound blocking material such as pockets, and can be robust to background noise.

chapter 4

Detecting Social Interactions using Speech Signals and Deep Learning in Smartphone Devices

4.1 Introduction

The work done in Next2Me [28] outlined the possibility of tracking social interactions in real time using a technique which can separate social groups that are close together but not part of the same interaction. It also specified how continuous sensing may be possible by using a duty cycling technique. However, there remains a limitation that all participants of a social interaction must carry a Smartphone device with them at all times and each Smartphone must have the Next2Me app installed and running in the background. In certain real world scenarios, perhaps this is not viable, and expecting all participants to install a continuous sensing application may not be realistic. Furthermore, in edge case scenarios such as meeting or cinema, the user might put their smartphone into airplane mode which would cause all access to the WiFi sensor to be lost. Although access to the microphone is still possible, without the context of WiFi signals, passive proximity detection between devices would no longer be available and the system would fail to perform. Therefore, it is paramount

in social sensing that the collection of data is less intrusive from the perspective of social interaction participants as a whole.

This section of the thesis outlines an alternative approach to Next2Me, through a system entitled *Speaking2Me*. Speaking2Me aims to allow the detection of social interactions using one smartphone only, targeting scenarios where metrics are needed to be obtained in real time about the socialness of a single-user. In order to do this, the system employs state-of-the art Deep learning techniques for speaker identification, to establish if a given speech sample recorded by a smartphone device belongs to a new or known speaker within a user's database of social contacts. By tracking the unique speakers contained within a historical database of voices, Speaking2Me can determine the count of speakers involved and match them with previous encounters. By doing this it can be used to create new ways of tracking personal socialness over time.

4.2 Motivation

Allowing people with sociability issues to monitor their levels of interactivity with other people can be of great value, and can help them to manage their condition. In particular, there is specific interest from people who suffer from social anxiety [128] or bipolar disorders [129], to detect social encounters and to understand how they might contribute in the development of their condition. Technical solutions for passive tracking of social interactions fall broadly into two categories: (i) instrumented environments with sensing capabilities that can track social interactions within the instrumented space [130], or (ii) mobile sensing of social interactions where all the relevant parties need to use a particular type of technology (e.g. wearable social tracking tags [106]). Both of these approaches have limitations that do not make them suitable for the continuous tracking of an individual's social life throughout their daily lives. The first approach only works in certain environments (e.g. workplace), while the second limits the data capture to social interactions between users who use specialised technology. Indeed, in order for an individual user to be able to track their own social life, technical solutions should work in any environment, without

the need for their social peers to use certain technology.

In this work we consider the detection of speech between individuals, where the participating parties can be accurately detected through the analysis of their voice patterns. Existing techniques in speaker identification tend to rely more on supervised machine learning techniques [7, 131–133], which do not scale well as they put a significant burden on the user to manually generate training data sets. In supervised learning, the fully-labelled data set can model which audio samples belong to which person, however, it already has the answer to that challenge. When a new speaker is introduced which the system was not trained for, it has to classify that sample as one of the existing labelled speakers.

In our work we consider the development of a speaker identification solution that can run on a single user's mobile device, and can accurately match voice samples with previously encountered individuals. Speaking2Me, is a system for unsupervised speaker identification using deep learning. It employs an autoencoder-based deep learning model, which lets the model free rein to find patterns of its own that can produce high-quality results. Unlike supervised techniques, there is no limit on the number of speakers which can be identified. This model can generate an *encoding* feature vector for every voice sample, which maintains close similarity with encoded vectors of voice samples of the same individual, and significant dissimilarity with voice samples from other people. The Speaking2Me system achieves high levels of accuracy in matching voice samples of people, without the need to pre-train the system with samples from the specific people.

4.3 Approach

The high level approach of our system was to use Deep Learning to generate embeddings which represent the voice of speakers. These embeddings would then be passed to a classifier to determine if the voice belongs to a new or existing speaker from a list of social contacts. An outline of the high level system design can be seen in figure 4.1.

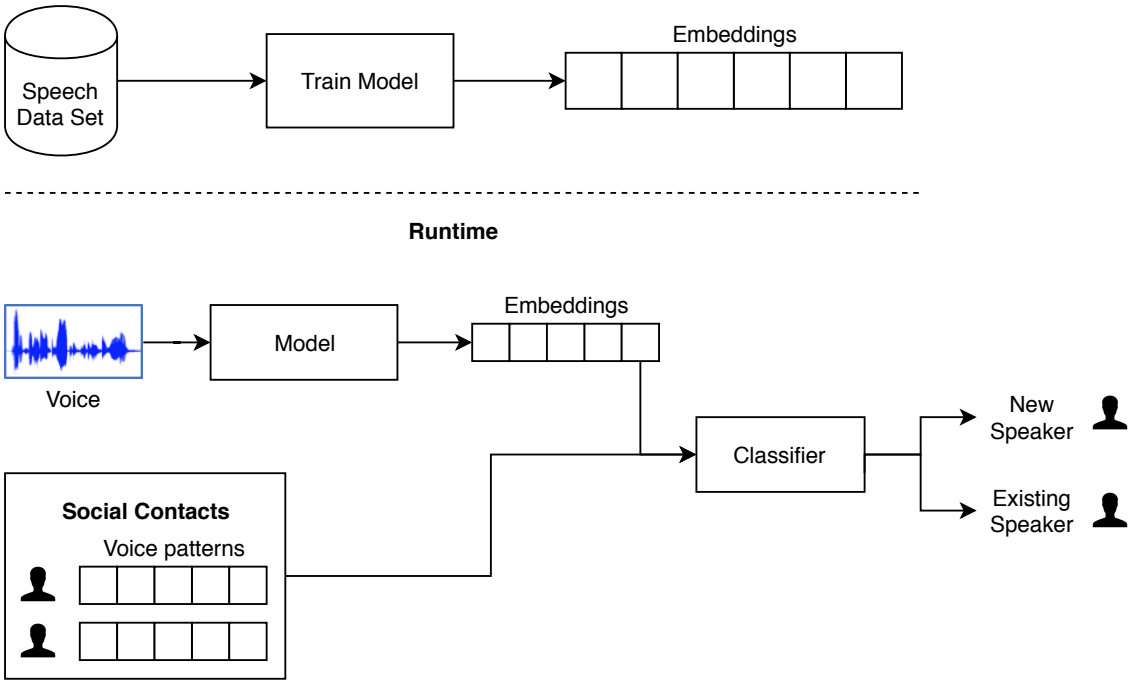


Figure 4.1: High-level approach of the Speaking2Me system. For training, a speech data set is used to train the model, which produces embeddings. At runtime, the voice is recorded and fed into the model where embeddings are produced. The embeddings are fed into a classifier which determines if the speaker is new or not. New speakers are enrolled into the list of social contacts.

4.3.1 Data Design

There are approaches that (1) capture data sets in the wild. In these settings, it is very difficult to build data sets that are sufficiently large enough for training data. There are also approaches that (2) use large public data sets, much more suited for a deep learning system.

In order to design and evaluate a system like this, we made use of public data sets to train models for speaker identification. However, in general these existing data sets are typically created in constrained conditions, with limited environmental diversity and are therefore limited in size. People might be reading from scripts and there is a lack of background noise, reverberation or overlapping speech. The most ideal data set would be recorded in a vast number of multi-speaker environments, riddled with naturally generated background noise and recorded using a variety of microphone devices.

4.3.2 Voxceleb Dataset

VoxCeleb [7] provides an automated pipeline to create data sets from open-source media, specifically YouTube, by using face-detection and face tracking to extract the audio of when people are speaking. The videos are shot in a large number of challenging environments, including predominantly multi-speaker acoustic environments, such as: red carpet events, outdoor stadium, quiet studio interviews, speeches given to large audiences, excerpts from professionally shot multimedia, and videos shot on hand-held devices. Crucially, the data contains noise from real-world background chatter, laughter, overlapping speech and room acoustics. Each speech file is more than 3s long and there are multiple samples for each speaker.

# of speakers	1251		
# of Male speakers	690 (55%)		
# utterances	153,482		
	Max	**Avg**	**Min**
# utterances per speaker	36	18	8
Length of utterances	145 sec	8.2 sec	4.0 sec

Table 4.1: Data set design of VoxCeleb. This table describes three entries in a field: maximum / average / minimum. An utterance refers to a sample of speech which lasts more than 3 seconds in duration

Due to the extensive and diverse contents of the VoxCeleb dataset, we considered that it would be a great training set for our proof-of-concept social sensing system. At the time of writing this, the VoxCeleb paper has already been cited 550 times, more notably used in highly cited work involving:

- Data augmentation that improves the performance of a deep neural network (DNN) embeddings for speaker recognition [134]
- A neural network-based system for text-to-speech synthesis that is able to generate speech audio in the voice of many different speakers, including those unseen in the training data [135]
- A neural architecture for directly processing waveform audio in speaker identification and speaker verification tasks [136]

The VoxCeleb dataset was constructed in 5 steps:

- Stage 1. To obtain a list of "Persons of Interest", VoxCeleb uses the list of people that appear in the VGG Face dataset [137], which is based on an intersection of the most searched names in the Freebase knowledge graph, and the Internet Movie Data Base (IMDB), of which approximately half are male and the other half female.

- Stage 2. The top 50 videos for each of the 2,622 faces are automatically downloaded using YouTube search. The word 'interview' is appended to the name of the person of interest in search queries.

- Stage 3. The HOG-based face detector [138] is used to detect the faces in every frame of each video downloaded from YouTube. Facial landmark positions are detected for each face. Within each detected shot, face detections are grouped together into face tracks using a position-based tracker.

- Stage 4. A two-stream CNN described in [139] which estimates the correlation between the audio track and the mouth motion of the video, to verify speech.

- Stage 5. Active speaker face tracks are then classified into whether they belong to the target speaker or not

An overview of this pipeline can be seen in figure 4.2.

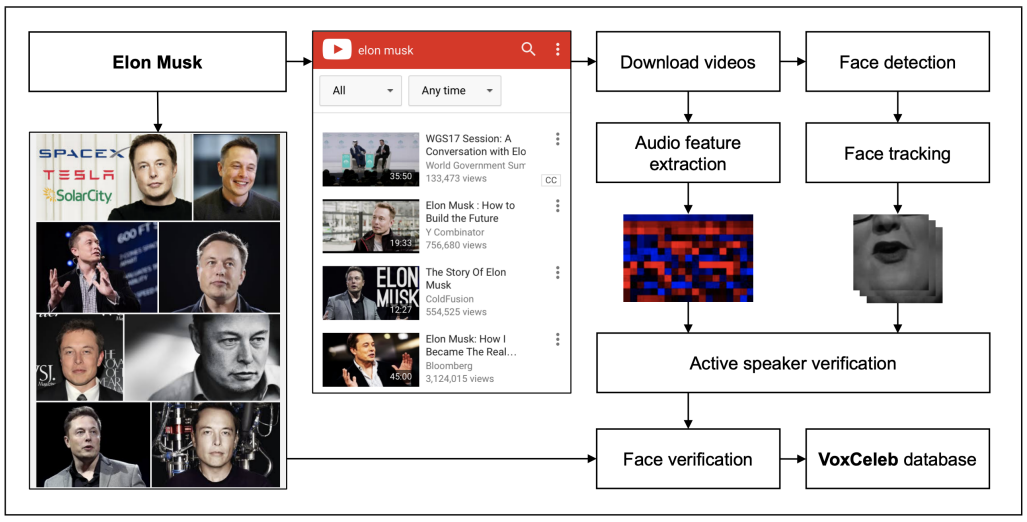


Figure 4.2: Data processing pipeline used to capture the VoxCeleb data set. This figure shows how celbrity "Elon Musk" is enrolled into their VoxCeleb database. Image Source: [7]

We used the VoxCeleb data set to build our Speaker Identification model, because of the large number of utterances taken from diverse environments, with real-world and multi-speaker noise. This is crucial to building a mobile-based speaker identification system, where mobile devices can be placed in a number of different environments, and there is typically challenging noise. Furthermore, testing a model against a 1251-class data set would provide enough diversity to prove that the model can work effectively with a large number of speakers, regardless of gender, age, ethnicity or language.

VoxCeleb describes it's own supervised methodology for speaker identification using CNN, which performs almost 20% higher than traditional state of the art baselines, and uses 67M parameters to achieve 80.5% accuracy across 1251 classes. The aim of our work is to develop an unsupervised Neural Network which does not require the enrolment of new speakers, and thus be more suitable for smartphone devices in the wild.

4.4 Feature Extraction

In order to build a speaker identification system, we first needed a set of features which represent the voice patterns of people's speech. This section describes the features selected for our speaker identification and speaker counting methodologies.

4.4.1 MFCC Extraction

The most important step in any speaker recognition system is to extract auditory features from a given audio input. Typically, Mel Frequency Cepstral Coefficients [140] (MFCCs) are used in these speaker recognition systems as they are described to accurately represent the envelope of the human voice. They can be described as perceptually motivated signal representations defined as the real cepstrum of a windowed short-time signal derived of the FFT of that signal. By introducing information about human perception, we can train systems using parts of the information which human listeners would find important. We specifically want to use these MFCC features to identify voice pattern for individuals.

In deep learning, some systems have proven to give good results by using a Mel-

Frequency Spectrogram as the input data [141]. However, such data is large in size and produces a vast amount of learnable parameters, which increases model size. For smartphone devices, it is optimal to extract a small amount of data, if possible, to provide a much more lightweight approach. Therefore, we looked into identifying the components of an audio signal which are best for identifying voice patterns whilst discarding background noise. Ideally, we wanted to build a model which would understand the general sound of a human voice, and not the contents of the actual speech. MFCCs are widely used in speaker tasks [142]. At a glance, this is how extracting an MFCC vector would work:

1. Before the spectral analysis, in order to amplify the energy in the high-frequencies of the input speech signal, a pre-emphasis filter is used:

$$H(z) = 1 - 0.95 \times z$$

2. Frame the input signal into small windowed frames of 25 ms

3. Apply a hamming window function to each frame, which can be described as:

$$w[n] = (0.54 - 0.46)\cos(\frac{2\pi n}{N-1})$$

   where, $0 \leq n \leq N-1$, and N is the window length.

4. From each frame, calculate an estimate of the power spectrum. This can be done by extracting an FFT and then calculating a Power Spectrum from it.

$$P = \frac{|FFT(x_i)|^2}{N}$$

   where, x_i is the i^{th} frame of signal x.

5. Apply the mel filterbank to the power spectra, sum the energy in each filter. This is a set of triangular filters that we apply to the power spectra.

   Firstly, we convert the upper and lower frequencies to the Mel-scale:

$$M(f) = 1125 \ln(1 + \frac{f}{100})$$

Then we can use the following to convert these to hertz:

$$M^{-1}(m) = 700 \exp(\frac{m}{1125} - 1)$$

Finally, we can create our filter banks. Once this is performed we are left with 26 numbers that give us an indication of how much energy is in each filterbank:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leqslant k \leqslant f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \leqslant k \leqslant f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

where m is the number of filters we want, and $f()$ is the list of $M+2$ Mel-spaced frequencies.

6. Take the log of all filterbank energies. This leaves us with 26 log filterbank energies

$$E[m] = log(|H_m(k)|)^2$$

for each frame a log-spectral-energy vector, $E[m]$ is obtained as the output of log energy computation.

7. Apply a Discrete Cosine Transform (DCT) of the log filterbank energies to extract the MFCCs. The standard way of calculating a cepstrum is:

$$C[i] = \sqrt{\frac{2}{M}} \sum_{m-1}^{M} E[m] cos\left(\frac{\pi i}{M}\left(m - \frac{1}{2}\right)\right)$$

where M is the chosen number of cepstral coefficients

8. Store the DCT coefficients between 2 and n, and then discard the rest

An overview of these steps is visible in figure 4.3

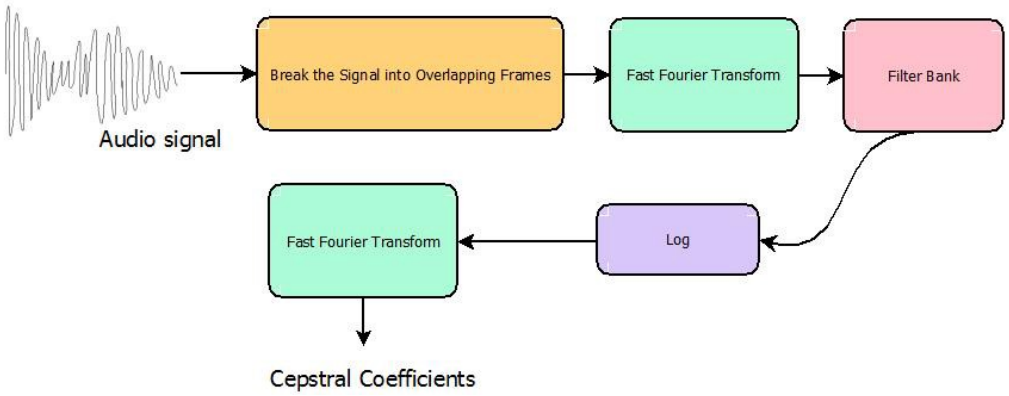


Figure 4.3: Overview of the steps to extract MFCC features from a speech sample. Here, Filter Bank refers to the mel filters (converting to mel scale) and Cepstral Coefficients are the MFCCs. Image Source: [8]

The MFCC feature vector describes the power spectral envelope of a single frame, however, it lacks the dynamical information about the speech, such as trajectories of the MFCC coefficients over time. Therefore, it is also common practice to append the *deltas* to the MFCC feature vector to increase speech recognition performance [143]. The deltas can be calculated as such:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{4.1}$$

where d^t is the delta coefficient calculated from speech frame t, computed from coefficients c_{t+n} to c_{t-n}. A typical value for N is 2. It is also possible to calculate the delta-deltas by applying the above formula to the delta coefficients, however, we wanted to obtain a lower dimension of features.

For our preliminary experiments, we used 16 MFCCs and their deltas, totalling to an input feature vector size of 32. This was an arbitrary and initial number of features that was selected for a preliminary exploration. It's important to note that the end-goal of the feature selection was to eventually optimise the number of MFCCs used, and we approached this in 4.6.2.1.

To make the training data and validation data sets comparable, we needed to apply scaling to the two sets. We did this by centering the data to make it have zero mean

and unit standard error. We used a library in Python for a scaler and fit the scaler to the training set of data. We then saved the scaler to disk and reused it to apply the same transformation to the validation set (and this should be used for newly obtained examples before forecasting).

4.5 Preliminary Experiments and Results

In order to understand the viability of using deep learning for speaker identification in devices with low resources, we first had to construct preliminary experiments with the VoxCeleb data set to understand the possibilities. The aim for this experiment was to replicate the design of a supervised model, similar to what VoxCeleb provides, whilst exploring techniques to reduce the memory footprint of the model. We undertook 2 main preliminary experiments (i) a 2D Convolutional Neural Network designed to classify short samples of speech and (ii) a 1D autoencoder Convolutional Neural Network designed to classify MFCC feature vectors belonging to speakers using distance metrics.

We decided to use a 2D Convolutional Neural Network because it appeared to work well in the literature. For example, the work in [144] investigated whether end-to-end learning for music audio is feasible using convolutional neural networks, specifically for music content tagging. In this work, the CNN is used with a spectrogram image to automatically learn features. The work in [145] investigated whether it was feasible to identify speakers using features generated by a CNN. The input of the model was a spectrogram of speech, similarly to the VoxCeleb's implementation.

Alternatively, we decided to also try autoencoders. Unlike 2D Convolutional Neural Networks, an autoencoder can be used to develop an unsupervised approach to a social sensing system. For example, the work in [146] uses an autoencoder for speaker recognition using a deep neural network, for removing reverberation, denoising and enhancing the speech present in audio samples to improve a speaker recognition system. Similarly, the work in [147] uses a deep autoencoder to denoise audio samples to enhance speech.

4.5.1 Architecture for 2D Convolutional Neural Network

Our architecture was based loosely on the model presented in [7], which was a 2D Convolutional Neural Network based on the VGG-M CNN [148]. We applied our own modifications the CNN in VoxCeleb to match a smaller input size and to provide a less computationally complex model. Table 4.2 outlines the model design and figure 4.4 shows an outline of the Neural Network process. Each convolution layer was followed by batch normalisation and a Rectified Linear Unit (ReLU) [149] layer. conv4 had support for 14 x 1, for the frequency domain, followed by an average pooling layer of 1 x n, where n depends on the length of the input speech. For example, for a 3-second input, $n = 23$. We used no max pooling due to the low input data size. The filter size of each convolution was 64, to maintain low parameter usage, with the exception of conv4 which used 128 filters. We used Dropout [150] across the network to reduce over-fitting. The output of the last layer is fed into a Softmax [25] in order to produce a distribution over the 1,251 classes. Using this model, we are able to reduce the learnable parameters from 67M in VoxCeleb to 3.1M; a parameter reduction of 95.4%

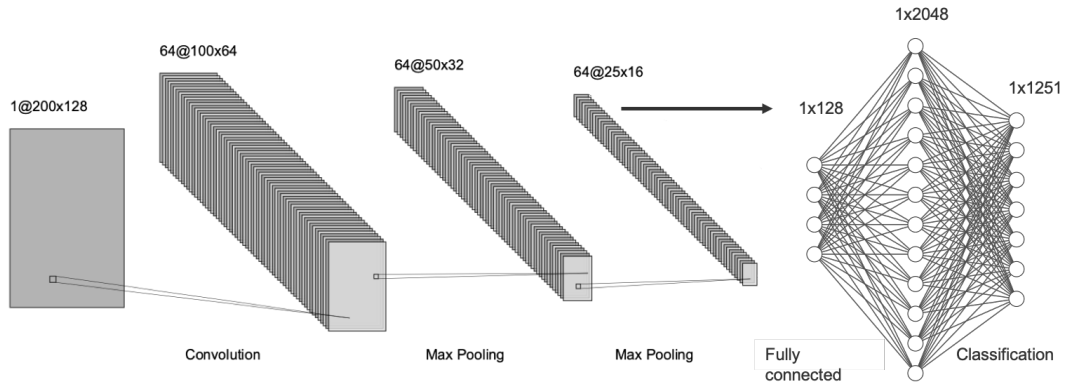


Figure 4.4: Neural Network diagram overview of the proposed CNN architecture. In this, multiple convolutions are passed into fully connected layers before being classified using a Softmax layer

4.5.2 Implementation

We used the Neural Network Toolbox from Matlab and trained on a NVIDIA Tesla K80 GPU. The network was trained using Batch Normalization after each layer. Dropout was configured experimentally, and outlined in figure 4.2. Other than this,

Layer	Support	Filter Dim.	# Filts.	Stride	Padding	Dropout
conv1	7 x 7	1	64	2 x 2	1	0
conv2	5 x 5	64	64	2 x 2	1	0
conv3	5 x 5	64	64	2 x 2	1	0
conv4	14 x 1	64	128	1 x 1	0	0.2
apool1	1 x n	-	-	1 x 1	-	0.2
fc1	-	128	2048	-	-	0.2
fc2	-	2048	1251	-	-	-

Table 4.2: CNN architecture for a 3-second input duration of speech. The input spectrogram is passed into 4 convolution layers before an average pooling layer, and then two fully connected layers

all default values of the toolbox were used.

4.5.3 Training and Validation

For training, we used the 138,361 full speech files for 1251 speakers from VoxCeleb training set and 6,904 files from the VoxCeleb validation set. The datasets here, are a collection of 1251 classes, where each class represents a person who is speaking.

For the training data, we used files where the duration was larger than 3s, and augmented the input by taking random crops of 3 seconds across the time-domain, specifically spectrogram crops of size 128 x 200, where 200 is equal to a 3-second duration. The training data was now 2D images of 128x200 in size, of spectrogram data, for 1251 classes. The validation data was processed in a similar way, and also consisted of 128x200 crops of spectrogram data, for the 1251 classes.

We trained the model using a learning rate of 0.05, and used no L2 regularization. Furthermore, the training data was shuffled before each training epoch, and the validation data was shuffled before each network validation to reduce variance and over-fitting. Zero-center Normalization was applied to the spectrogram image input layer. The training was validated by the Means Squared Error (MSE) calculation, which can be described as:

$$MSE = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} (observed - predicted)^2$$

MSE measures the average squared error of the predictions made by the model. It

calculates square difference between each point of the predictions and the target and then produced an average of those values. The higher this value is - the worse the model performs.

During the training of our model, we used the MSE loss function to monitor the performance of the model as it trains. This can be seen in figure 4.5, which shows how the MSE loss had decreased over multiple epochs during the training process.

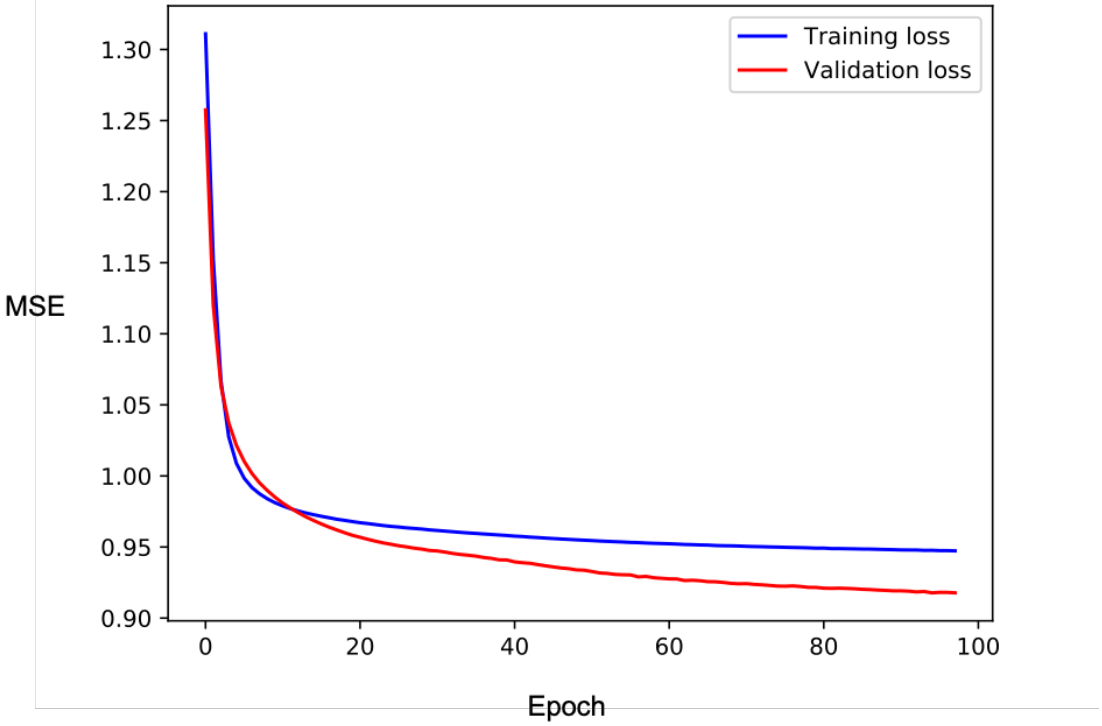


Figure 4.5: Training and validation loss values captured over multiple epochs during the training progress. A lower MSE loss represents a model which should perform better.

4.5.4 Testing

We used a separate data set and used it for testing the performance of our network. To build this new data set, we took the full-size files from the VoxCeleb 'test set' section, which is a separate data set of 4,874 audio files from the 1251 speakers. The files have no crossover with the validation or training sets. From these files, we take the speech inputs and split them into 3s fixed-length segments. These segments were classified using the trained model, and then the Softmax scores were averaged to give a final class prediction for a full-size speech input. This follows a similar methodology to VoxCeleb's "CNN-fc-3s" architecture, which performed at 72.4% accuracy. We

used the validation data to estimate the model's performance after each training epoch, and used an implementation of Early Stopping to stop the training procedure once the validation accuracy has failed to improve after 5 epochs.

We evaluated the performance of this 2D CNN Model architecture over the VoxCeleb database, which consisted of the 1251 classes. The results of this are shown in table 4.3. In these results, we analyzed two classification methods. The first, "full-file" follows the classification methodology in VoxCeleb by splitting a variable-input image into 3-second segments, and providing a Softmax score for each segment. The 3s Softmax scores for each variable-input image are then averaged to provide a $Softmax_{avg}$ which was used to provide classification for an entire full length audio file. The second classification method also splits a variable input image into 3-second segments, and then provides an accuracy for the single 3-second input.

4.5.5 Results

Architecture	**Full-File F-Score**	**3s Segment F-Score**
CNN-fc-3s no var. norm	68.22%	75.79%

Table 4.3: Table of results for Speaker Identification using CNN-fc-3s architecture on the VoxCeleb dataset

Looking at the results in table 4.3, we can see that there is significant scope in reducing input features and still maintaining an acceptable accuracy for the classification task. For full-file F-Score metric, it performs well compared to VoxCeleb's 72.4% accuracy, however, the *3s Segment Accuracy* result for this preliminary experiment shows how well the trained model can identify independent small samples of speech without the need to classify entire tracks. In conclusion, it is clear that we can learn a lot without replying large complex models, which brings around viability for a mobile-based neural network. However, classification tasks are closed-set and would not work well in real-word scenarios when new speakers are entering conversational databases. In our next preliminary experiment, we looked into the feasibility of an unsupervised approach.

4.5.6 Autoencoder Neural Network

An autoencoder [151] is a transformation applied to an input vector that tries to map samples from one space into the same space. More specifically, an autoencoder will

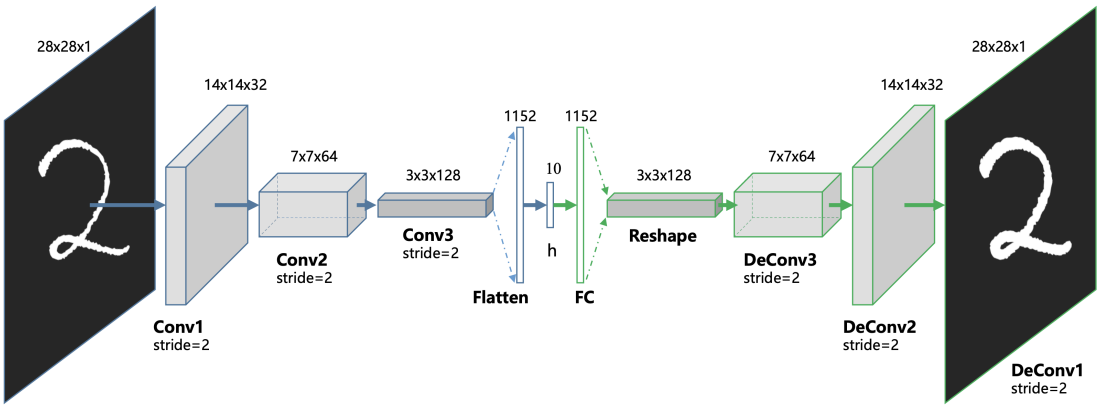


Figure 4.6: Overview of the autoencoder example for MNIST, illustrating how the autoencoder can learn an embedding to represent individual hand-written letters. Image source: [9]

try to squeeze the high dimensional space through a lower dimensional bottleneck (known as the encoder) to learn how well it can reconstruct the original samples (known as the decoder). Figure 4.6 [9] shows an example structure for a convolutional autoencoder for MNIST (a large database of handwritten digits that is commonly used for training various image processing). In the middle of this figure (h), there is a fully connected autoencoder whose embedded layer is composed of only 10 neurons. The rest are convolutional layers and convolutional transpose layers, to be trained directly in an end-to-end manner. In this example, we can take images of a handwritten number '2' and train the autoencoder to construct an embedding at h which represents the number 2. We can then extract and use this embedding to classify which handwritten letters are '2' and those which are not.

In our case, we use the encoder and the decoder network as a black box that handles the transformation. With our 1D feature vector inputs, we can imagine that this will work a lot like lossy image compression, where information is lost as the dimensionality is reduced, but the original image can still be reconstructed with only a small loss of quality. In unsupervised speaker identification, the during runtime the output of the encoder is not actually of any practical use. Instead, we care more about the hidden representation that the autoencoder has learned. We can use this learned representation as a *fingerprint* to distinguish some speakers from others. The outline of an autoencoder for our use case can be seen in figure 4.7. In this figure, the 32 MFCC input is passed through a series of layers g_ϕ, which encodes

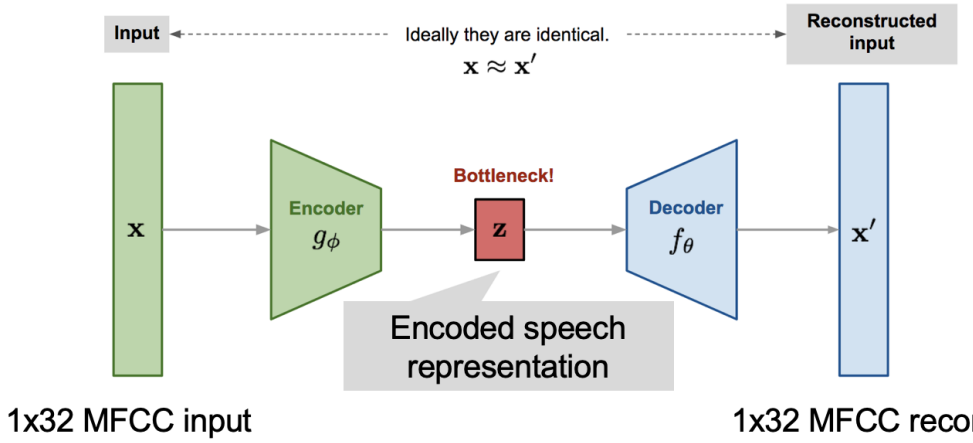


Figure 4.7: An overview of the autoencoder example, for a use case involving MFCC features as an input. Here, the 1x32 MFCC input is encoded to the bottleneck layer. The encoded speech representation from the bottleneck layer is then decoded to reconstruct the original input.

the input into the bottleneck z. The bottleneck encoded representation of the input is then reconstructed through the decoder layers in f_θ. The reconstructed output is then compared to the original input, and a loss function is used to calculate the differences. Ideally, the reconstructed output would be identical to the original input, meaning that a sample of MFCCs can be encoded and decoded with barely any loss. The bottleneck encoded representation can then be used for classification.

4.6 Model Architecture and Design

Concluding from the preliminary analysis of deep learning network types, it became evident that an autoencoder was the best choice, particularly for a unsupervised approach. Considering that mobile devices would require a lightweight model architecture, an autoencoder makes a good choice because it works by encoding it's features into less dimensions, resulting in a smaller feature vector size. Moreover, autoencoder generally requires fewer learnable parameters than a typical Convolutional Neural Network architecture for classification tasks.

We made some different design choices to fit this problem and to ensure that we're learning voice patterns and not the content of speech:

(a) We use longer windows, specifically 12-seconds, to ensure that the network does

not focus on the speech contents, but instead the voice characteristics which occur over longer periods of time. Practically, the input to the autoencoder is the set of MFCCs calculated over a longer window of speech and therefore obfuscates the actual speech content.

(b) For the VoxCeleb training set, we also constructed a collection of pairs for each voice sample. These pairs are from the same speaker, but from a different voice sample. We wanted to make sure that when the autoencoder reconstructs the input, it calculates the loss based on the pair and not the original sample. By doing this, each input is reconstructed to different speech from the same speaker. We do this to ensure that whilst training, the network captures encoded vectors which represent *how* the speaker speaks, and not the contents of the speech.

4.6.1 Initial Autoencoder design

We created an initial 1D Convolutional Neural Network autoencoder architecture that would be used to experiment with feature input size, window size, and model optimisation. According to our preliminary work which identifies that a 1D CNN autoencoder performs well with MFCC features, we decided to base our model architecture around a deep 1D CNN autoencoder with a maximum of 2 Convolutional layers in depth to reduce computational cost. This model's diagram can be seen in figure 4.9, with an outline of each layer in table 4.4. Figure 4.8 provides a neural network diagram of the proposed autoencoder network. Each Convolutional layer used Rectified Linear Units (ReLU) activation (since ReLU is commonly used as an activation layer in deep learning) except for the output layer, which use a Sigmoid activation function. After each activation layer, we applied Max Pooling in 1 dimension with a size of 2, so that it would encode the data by 50% each time we applied it. The output of the second Max Pooling layer (max_pool2) was used as our encoder output.

4.6.1.1 Data Set Split

We partitioned the VoxCeleb dataset into three partitions: training, validation and test sets. The split was kept the same as VoxCeleb, and the same data set as we described in 4.5.3. An overview of the data set contents are described in table 4.1.

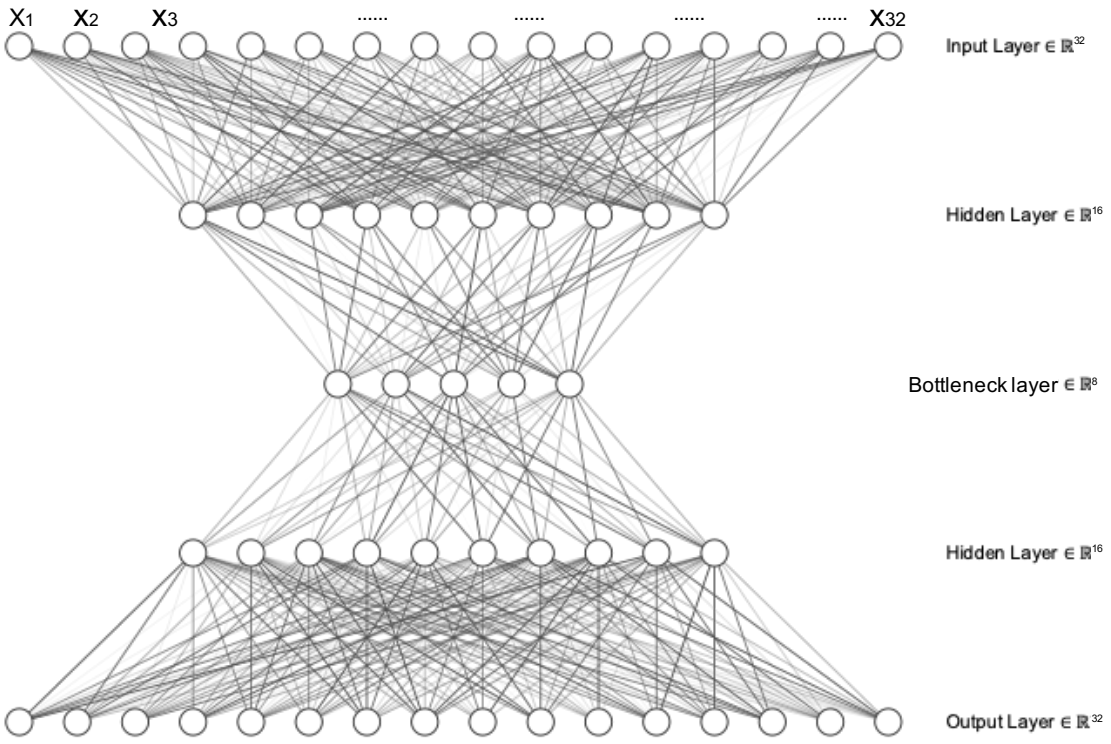


Figure 4.8: Proposed initial 1D CNN architecture for a 12-second input duration of speech represented as a Neural Network diagram. In this figure, the MFCCs are inserted from x_1 to x_{32} and encoded into the bottleneck layer. The encoded features are then extracted from this bottleneck layer. The encoded representation is then decoded to the output layer

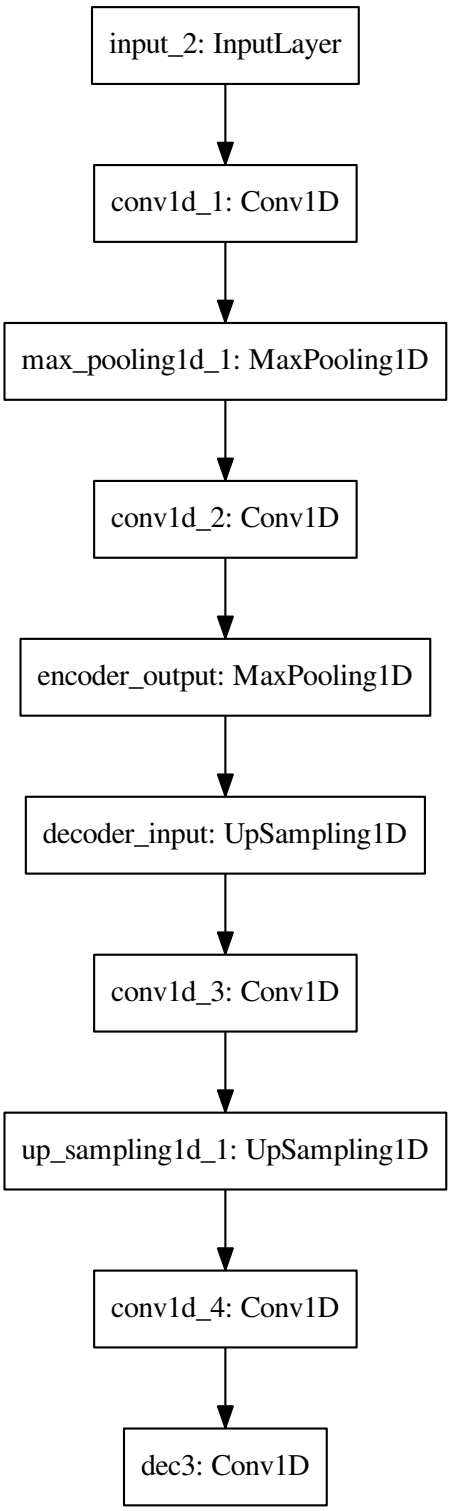


Figure 4.9: Overview of the initial model used during the preliminary experiments in 4.6.1. The labels are in format of layer_name: layer_type

Layer Name	Layer Type	Support	# Filters	Activation	Output size
input	Input Layer	-	-	-	32
enc_conv1	Convolutional	3	16	ReLU	32
max_pool1	Max Pooling	2	-	-	16x16
enc_conv2	Convolutional	3	8	ReLU	16
max_pool2	Max Pooling	2	-	-	8x8
upsampling2	Upsampling	2	-	-	16x8
dec_conv2	Convolutional	3	8	ReLU	16
upsampling1	Upsampling	2	-	-	32x8
dec_conv1	Convolutional	3	16	ReLU	32x16
output_conv	Convolutional	3	1	Sigmoid	32

Table 4.4: Proposed initial 1D CNN architecture for a 12-second input duration of speech. Layer max_pool2 represents the output of the encoder and output_conv represents the output of the decoder

For the preliminary experiments of Speaking2Me, more specifically, we chose to use a small subset of VoxCeleb dataset, where only the Person of Interest (POI) whose name started with an "E" was selected to be trained and validated. This subset of POIs provided enough diversity between male and female speakers, and is specifically used in the Voxceleb paper as a subset for testing their trained model. By reducing the number of POIs, the preliminary experiments of Speaking2Me could be performed much quicker and thus eliminated the time limitations involved in machine learning with large data sets. Testing involved variable amount of speakers, from the remainder of the VoxCeleb data set (POIs whose name does not start with "E"). For the final model training, we used the entire 1251 classes from VoxCeleb for the training, validation and test sets.

4.6.2 Input Features

For input features we chose audio samples which were mono, single-channel 16-bit stream at a 16 kHz sampling rate. This allowed all of the speech frequencies between 300 Hz and 3400 Hz, and had a good balance of audio quality. Any audio file which was stereo would first be converted to a single-channel format. For feature selection, we picked the Mel-frequency cepstral coefficients (MFCC) as the most appropriate input for speech processing. We sampled audio as a 12 second window size (as described in section 4.6.2.2) with a 50% overlap sliding window to ensure that we are encoding features which represent the voice of a speaker, and not the contents of

the speech. 16 MFCCs were extracted from the 12 second window and their deltas, to create an input vector of 32 in size.

4.6.2.1 Picking the Best Number of MFCCs

Picking the number of MFCCs for the input vector of a deep learning model can be a trivial task. To provide justification, it was deemed important to produce experimental results for different MFCC feature vector sizes to determine what the best amount was. To achieve these results, we followed the preliminary experimental set-up of using a 12 second window size with 50% slide and extracted the MFCCs and their deltas from the audio track. The number of MFCCs for a given experiment can be seen in figure 4.10 and the total feature size was the **number of MFCCs * 2** due to the addition of the deltas.

The accuracy was calculated using a Distance Classifier which uses Euclidean distance to calculate the distance between the encoded feature vectors and compares the distance of these against all other speakers. The encoded feature vectors are what we extract from the autoencoder at the bottleneck, which is an encoded representation of speaker's voices. We then use the smallest Euclidean distance to classify each sample as the person who was most likely talking. This is described more in section 4.7.1.

4.6.2.2 Picking the Best Window Size

We established the best window size by running the initial 1D CNN with data created by a different window sizes. We run the training, validation and testing multiple times and make note of the accuracy produced by the distance based classification described in section 4.6.2.1.

Looking at figure 4.11 we can see that the accuracy increases as the window sizes increases. It is likely that this is happening because the MFCCs in shorter windows represent more about *what* people say and not *how* they say it. So as the window size increases, the feature vectors generated are more suited to a speaker identification task. In our final autoencoder design, we will use 12-second windows.

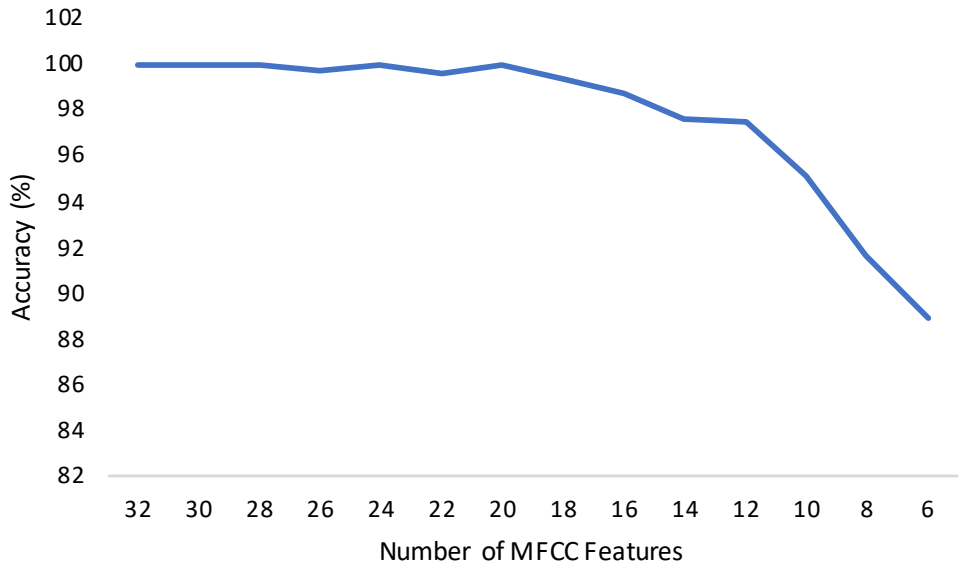


Figure 4.10: Comparison of distance based classification (Euclidean distance) for validation accuracy produced by variable MFCC features as an input vector. Here, as the number of MFCC features decreases, the classification accuracy also decreases

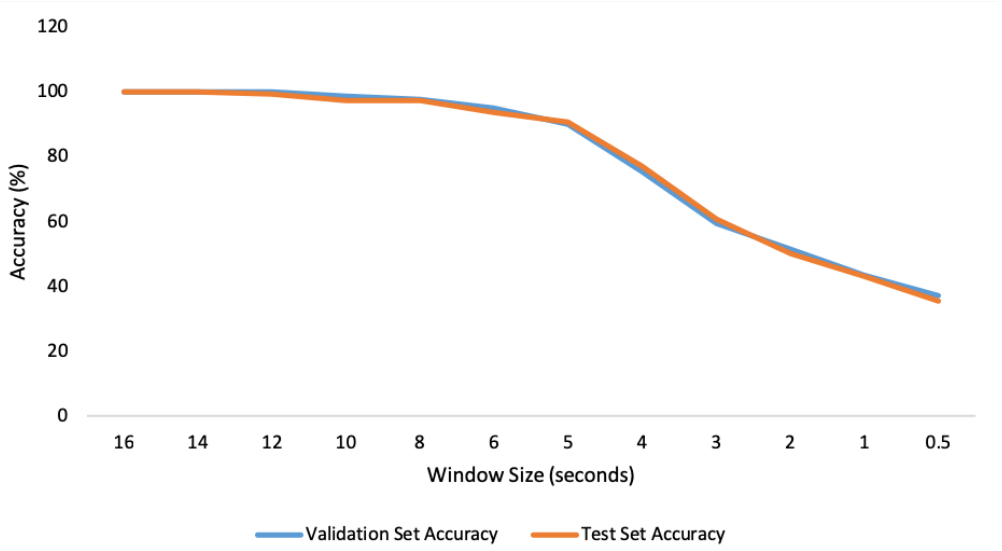


Figure 4.11: Accuracy of the system for variable window sizes using a Distance Classification technique. Here, as the window size duration gets shorter, the validation accuracy and the test accuracy decrease. The validation accuracy refers to the accuracy produced whilst training against a separate validation data set. The test accuracy refers to the accuracy produced from a completely separate testing data set that is not used during training.

4.6.3 Final Autoencoder design

In our preliminary experiments, we experimented with a number of different network architectures before choosing a 1D CNN Autoencoder due to its unsupervised nature

and yields good results when using a distance-based classification technique.

The output of our preliminary experiments allowed the final autoencoder design to be determined through experimentation with the number of MFCC and Deltas needed for the input vector and size (in seconds) for the window of the audio sampling. As such, we chose to extract a 12 second window size with a 50% overlapping sliding window (identified experimentally in 4.6.2.2 to achieve high accuracy in speaker identification). 20 MFCCs were extracted from the 12 second window and their deltas, to create an input vector of 40 in size (identified experimentally in section 4.6.2.1). It is our hypothesis that we can improve the performance of a 12-second window by using Hyperparameter optimisation, which we describe further in our Discussions chapter (5).

4.6.4 Autoencoder Validation

During our training, we use the MSE loss function to monitor the performance of the model as it trains, in the same way as is described in section 4.5.3. The input is a 12-second speech from a speaker, represented as a feature vector. This input is encoded by the autoencoder and then reconstructed. The reconstructed output is then validated against it's pair: a 12-second speech from the same speaker, but from a different audio file. The model loss is calculated by the MSE different between the reconstructed input, and the pair, as such:

$$MSE = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} (r - p)^2$$

where r is the reconstruction of the input and p is the pair assigned to the original input.

4.7 Classification Methods

We built an autoencoder model which will produce encoded representations of peoples voices. We needed to build a classification technique for discovering known encountered voices vs new people. We used the encoded vectors produced by our autoencoder model against the VoxCeleb training and validation data sets. The

autoencoder produces encoded representations of MFCC vectors for 12-seconds of audio. The training encoded representations consisted of 163,555 samples across the 1251 classes, and the validation encoded representations consisted of 7,972 samples across the same 1251 classes. We present two techniques for classification:

- A Closed-Set Distance-based Classification method
- A Binary Classification via a trained Neural Network

4.7.1 Closed-Set Classification

To test the performance of the trained model, we developed a Distance Classifier which uses Euclidean distance to calculate the distance between the encoded feature vectors and compares the distance of these against all other speakers. The encoded feature vectors are what we extract from the autoencoder at the bottleneck, which is an encoded representation of speaker's voices. We then use the smallest Euclidean distance to classify each sample as the person who was most likely talking. Euclidean distance can be defined as:

$$distance = \frac{1}{n}\sum_{i=1}^{n} |vec^a - vec^b|$$

where n is the number of elements within the vector of features produced by the autoencoder, and vec^a and vec^b are the output vectors produced by the autoencoder for speech samples a and b.

The accuracy was provided by this classification technique was output as 81.7% on the test set, an improvement on the supervised approach from VoxCeleb. However, this was assigning encoded vectors to 1251 classes, which is closed set and won't work in an unsupervised manner. The system would likely need to be retrained whenever a new class is introduced. To avoid this, we instead re-approached the challenge with an unsupervised binary classification technique.

4.7.2 Binary Classification

The idea was to design a model which would take the encoded representation for pairs of speakers and insert them into a system that would classify each as 'same' or 'different' speakers.

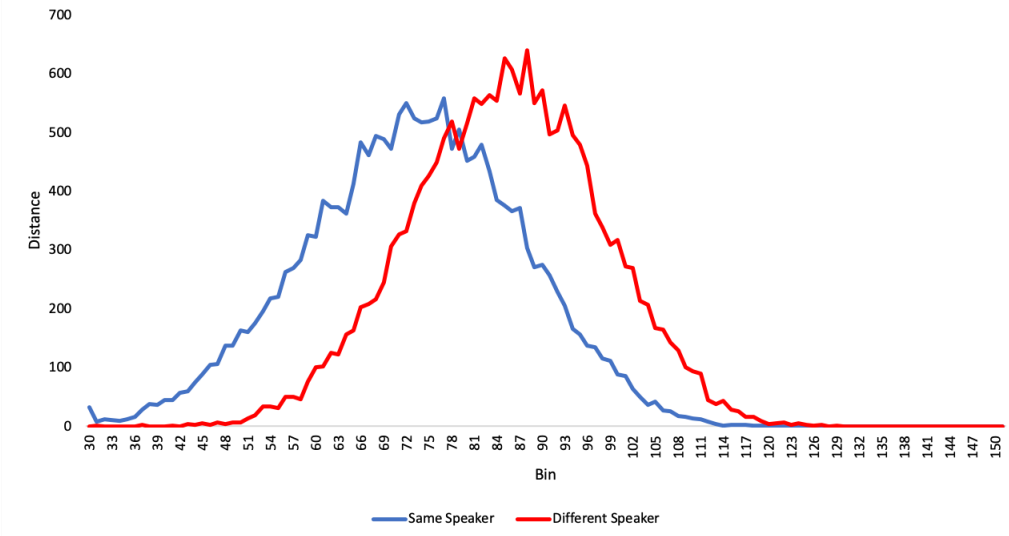


Figure 4.12: Distribution of Frequency Bin Euclidean Distances between Speakers who are the same and Speakers who are different.

In this work, we presented two techniques:

1. A threshold-based classification, using euclidean distance
2. A Logistic Regression classification technique

It was our hypothesis that the threshold-based classification would yield the lowest accuracy, whereas a logistic regression based method would provide better results because it might be able to learn from the encoded representation pairs vectors themselves.

4.7.2.1 Threshold-based Classification

In this method of classification, we try a technique of classifying between "same" and "different" speakers by using a threshold of Euclidean distance. This would work by calculating the Euclidean Distance between two samples and then noting if the ground truth labels are from the same speaker. In order to calculate the correct threshold, we used the VoxCeleb test set samples and calculated the Euclidean Distance between every possible combination of audio samples, and recorded the Euclidean Distance for when speakers are the same and when they are different. We plot the results of this in figure 4.12.

Looking at the figure 4.12, we can determine that the best threshold is the point

where the two lines cross over. However, it's quite clear that if we applied this threshold, the results would be poor; the distributions overlap too much and simulate that the accuracy would likely be poor for this method of classification.

Instead of continuing with a threshold based approach, we decided to try a Logistic Regression Classification techniques, in hopes to boost accuracy by letting the network learn from the samples themselves instead of relying on a fixed threshold.

4.7.2.2 Logistic Regression Classification

To construct a new dataset suitable for binary classification, we took the encoded representations for training and validation, respectably, and iterated across the entire collection. For each sample in the collection (source samples), we allocated 2 randomly selected samples from the same class, and 2 randomly selected samples from a different class (target samples). We append the target sample to the source sample to create a new data entry - where the left half of the vector is the source sample and the right side of the vector is the target sample. We also assign a label value to each - if both the source and the target are from the same class, then the label is 1, however, if the source and the target are from different classes then the label is a 0. This means that we construct a new dataset of speaker pairs where the speakers are either the same or different speakers, and each training or validation sample is assigned 4 randomly selected pairs.

For the training set, 163,555 initial samples were transformed to 654,220 samples, where each sample contains a pair of encoded representations of either 'same' or 'different' speakers (2 classes). For the validation set, 7,972 initial samples were transformed to 31,888 samples, as seen in table 4.5. For testing, we took the 4,874 files for 1251 speakers, from the VoxCeleb test set. We also used the VoxCeleb trial pairs for Verification, which is a list of 37,500 pairs of speech files, with ground truth of whether or not the pair is from the same speaker. The trial pairs are designed to test the performance of a speaker identification system. In our system, we test the binary classification performance based on the trail pairs, as such:

1. For each pair, split into 12-second windows

2. For each window, extract the MFCC input features

3. For each pair, insert the input features into the autoencoder to generate the encoded representation

4. Combine the encoded representations generated by the pairs into the same vector

5. Classify the pairs as *same* or *different* by inserting the combined vector into a binary classifier

# of speakers	1251
# training samples	654,220
# validation samples	31,888
# test samples	4,874
# test pairs	37,500
# classes	2

Table 4.5: Data set overview for the binary classifier. The classes here, are "same" or "different" speakers. From 4,874 test samples, we produce 37,500 audio pairs of same/different people speaking

We designed a binary classification model which would take an input sample and learn whether the sample contains two encoded representations of the same speaker, or different speakers.

4.7.3 Model Architecture

Layer	**Units**	**Activation**
dense1	512	PReLU
dense2	256	PReLU
dense3	128	PReLU
dense4	64	PReLU
dense5	32	PReLU
dense6	2	Softmax

Table 4.6: Final Binary Classification Model. Each layer is densely connected and uses decreasing units per layer. Each layer uses PReLU activation. Classification is performed by the final softmax layer (for 2 classes)

For the model architecture skeleton of the Binary Classification, we used simple densely-connected layers [152] due to their ability to be substantially deeper and more accurate (the final model is described in table 4.6). Our model is designed

to take the input (512 x 1) and reduce the dimensions by half for each layer. The final layer would then be 2 units, for binary classification. We used Hyperparameter Optimization to choose the parameters for us.

4.7.4 Hyperparameters

We designed an automated way to pick the best parameters for our new model using Talos and Keras. We decided that we wanted to focus on the following parameters for this:

Parameter Group	**Value**
First Layer Activation	ReLU
	PReLU
	LeakyReLU
	eLU
	Tanh
Optimizer	Nadam
	Adam
	RMSProp
	Stochastic Gradient Descent (SGD)
	Adagrad
Learning Rate	1.00^{-2}
	1.00^{-3}
	1.00^{-4}
Batch Size	Random range from 64 to 2048

Table 4.7: Outline of the parameters we investigated for the Binary Classifier Hyperparameter Optimization

We ran the Hyperparameter Optimization over 311 permutations of the model, where each permutation would train the model until Early Stopping determined that the validation loss could not improve for 3 epochs. The validation loss used binary cross entropy, which can be described as:

$$loss = -(y \log(p) + (1 - y) \log(1 - p))$$

where y is the binary indicator (0 or 1) if class label c is the correct classification for observation o and p is the predicted probability observation o is of class c.

We took the results of the Hyperparameter Optimization (outlined more in chapter 5) and sorted by the lowest validation loss, to pick the best performing model. This model trained for 12 epochs, used PReLU activation for each layer, required 2048 batch size and 0.001 learning rate. The optimizer used was RMSProp. The best model architecture is outlined in table 4.6.

4.7.5 Results

We used the parameters determined by the Hyperparameter Optimization to build our model, and then trained and validated the model from the data set described in section 4.7.2.2. The trained model performed with 80.43% F-Measure on the training data set and 84.21% F-Measure on the validation data set. This measurement of performance was calculated using Keras, which tests the performance of the trained model against a new data set. Specifically, we obtained our result by testing the model against the training set used, and then with the validation data set, which is a data from speakers not used in training.

We used the VoxCeleb trial samples (as outlined in 4.7.2.2) to test the performance of the entire sensing system. These trial samples are a list of 37,500 pairs of speech files, with ground truth of whether or not the pair is from the same speaker or not. We extracted the encoded representation from each 12-second window from each trial pair using our autoencoder model, to construct multiple inputs for each trial pair where each input represents 12 seconds of speech for both speakers as a combined vector. We insert this input into the binary classifier. We calculate the precision of our binary classifier on the trial pairs as 74.72%. The precision can be defined as:

$$precision = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where TP are the true positives (the samples that were identified correctly) and FP are the false positives (the samples that were not identified correctly).

		Predicted		
		Same Speaker	Different Speaker	Total
Ground Truth	Same Speaker	13988	4917	18905
	Different Speaker	4536	14059	18595
	Total	18524	18976	37500

Table 4.8: Confusion matrix showing the breakdown of the precision produced by testing the system against the trial pairs

4.8 Scalable Conversation Generation

To further test the performance of the Speaker Identification System, we looked to test the system against real-world data, however, real-world data is difficult to gather in large quantities. Instead, we worked towards building a small system for generating conversations using speech from people recorded in real and diverse environments.

Our conversation generation system accepts an input data set of labelled audio. The system then accepts two configurable parameters: number of speaker for the conversation and number of conversational turns (for example, the person speaking will switch n times). Based on the configuration, from those n people, we pick all of their audio to insert into a conversation. We then build a timeline of a audio, where we pick a random talking duration between 4 and 20 seconds and insert that into the conversation timeline until the configured number of conversational turns has been satisfied.

The system randomly generates a conversation of audio like so:

In algorithm 1, the conversation is generated by looping through the duration of the desired conversation and selecting random duration of speech to insert at given intervals. Each speech sample inserted is treated as a conversational turn (the moment someone else starts to speak) and the next speaker is anyone from the collection of random n speakers, except for the previous speaker. The output of algorithm 1 is a timeline of speech by n speakers where each speakers talks for a

Algorithm 1 Conversation Timeline Generation

1: Load data set and labels
2: Randomly select n speakers from data set into *speaker_collection*
3: Initialise *timeline* as an empty list
4: **repeat**
5: Randomly select a speaker from *speaker_collection*
6: Randomly choose a duration of speech between 4 and 20 seconds
7: Extract speech sample for given duration which doesn't exist in timeline already
8: Append extracted speech to *timeline*
9: **until** max number of conversation turns

random duration of time.

4.8.1 Conversation Speaker Identification Performance

To calculate the performance of the Speaker Identification system, we used our methodology on 100 generated conversations and calculated the precision (using algorithm 2) of each conversation via the algorithm 1. The average precision was then found from the 100 conversations.

In these tests, we built random conversations using 20 conversational turns and 3 participants per conversation. This means that the conversations generated would be between 1:18 minutes and 6:36 minutes long, and conversations would be 3:57 minutes on average. We ran the Conversation Generator 100 times and then determined the average precision for all experiments, and the average speaker count across the 100 conversations. Each conversation generated was random, and was likely to be significantly different to other generated conversations. Each conversation can pick from a selection of 40 speakers from the VoxCeleb data set (participants whose voice starts with 'E'), and use audio samples from the VoxCeleb testing set.

Algorithm 2 works by enrolling new speakers into an enrolment database. It determines if a speaker is new, when no matches can be found for a given sample when testing against the enrolment database. Matches are determined when a given sample is classified as the same speaker within the existing enrolment database with a probability of more than 0.5; when no enrolled speakers can match with a probability higher than 0.5, that sample represents a speaker who is new. We then determine speaker count by the size of enrolment database. An overview of the system can be

Algorithm 2 Unsupervised Speaker Identification Algorithm

Load data set and labels
Randomly select n speakers from data set into *speaker_collection*
Set *correct* to 0
Initialise *timeline* as an empty list
Initialise *enrolled* as an empty list
repeat
    Randomly select a speaker from *speaker_collection*
    Randomly choose a duration of speech between 4 and 20 seconds
    Extract speech sample for given duration which doesn't exist in timeline already
    Append extracted speech to *timeline*
until max number of conversation turns
repeat
    Take 12-second subset from the timeline as *speech*
    Extract the MFCCs from *speech* as $MFCCs$
    Extract *encoded* representation from $MFCCs$ using autoencoder
    if *enrolled* is empty **then**
        Insert *encoded_sample* into *enrolled*
    else
        Initialise *matches* as empty list
        for each *voice_sample* $\in$ *database* **do**
            Take the stored *encoded_sample*
            Combine with other *encoded*
            Insert new vector into Binary classifier
            Extract probability of both encoded samples being from the same speaker
            if probability > 0.5 **then**
                Insert probability into *matches* with *speech*'s label
            end if
        end for
        if matches is empty **then**
            Insert *encoded_sample* into *enrolled*
        end if
    end if
    Set *best_match* to the label which appears most in *matches*
    if *best_match* is *speech* label **then**
        Set *correct* to *correct* + 1
    end if
    Move forwards on the *timeline* by 12 seconds
until end of timeline
print: $\frac{correct}{total\_samples} \times 100$

found in figure 4.13.

The performance of our Unsupervised Speaker Identification algorithm is 74.72% average precision over the 100 generated conversations.

4.8.2 Speaker Counting

The speaker count is the size of the enrolment database. This means that speaking counting works in the following way:

1. Extract features from audio
2. Run Speaker Identification methodology on the extracted features
3. Estimate a count of speakers within the conversation based on the output of the Speaker Identification system

To estimate the count of speakers, we took the size of the enrolment database for all of the 100 experiments and calculated the average. In the above experiment, the speaker counting found 3.4 speakers on average.

4.9 Conclusions and Future Work

In this work, we showed the Speaking2Me system which works by extracting encoded representations of people's voices and classifies them using a trained binary classifier. The trained autoencoder model is 47kb and the binary classifier model is 4mb in size, both of which can easily be stored on a user's smartphone device. We tested the performance of the entire system through multiple auto-generated conversations to achieve average precision of 74.72%; similar to the supervised approach that VoxCeleb used.

Although the system has not been tested in the wild, the diversity of the data set used (including voices in noisy environments) indicates that this is a robust solution. For future work, we will organise user studies where participants attend set up meetings and other social scenarios with a data collection app installed on their smartphone devices.

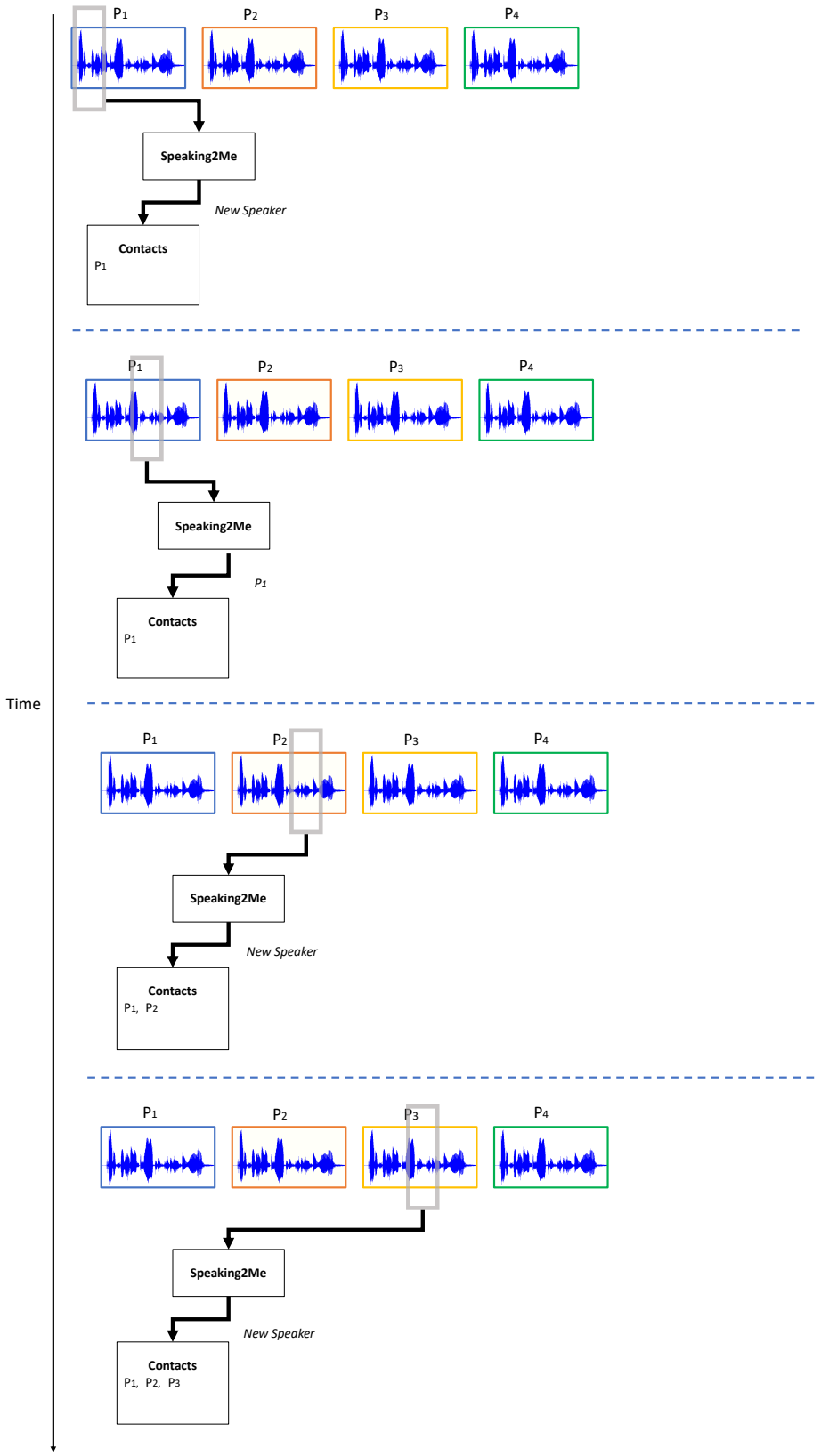


Figure 4.13: Overview of the Speaker Identification System. This figure shows how samples of audio can be extracted and classified using the Speaking2Me system to produce a collection of social contacts. The y-axis is time. As time moves forwards, the system can detect 3 social contacts - p_1, p_2 and p_3 where p_n is a participant from a social interaction

CHAPTER 5

Discussions

This thesis explored the sensing of human social interactions using smartphone devices, comprised of two principal contributions:

- A collaborative sensing system designed for Android devices entitled 'Next2me', which aims to capture the social interactions performed by social groups within close proximity to each other, but separate by nature. The system is deployed as a smartphone app that users can install onto their Android devices, and works by automatically collecting data about the signal strengths of nearby WiFi access points to determine if people are co-located. Once the system determines that people are co-located, a cloud service uses the similarity of *audio fingerprints* captured by the smartphone microphones to separate social groups. This system does not require the training on voice samples of participating users and yields high precision in these dense social scenarios.

- A separate and scalable continuous tracking system entitled 'Speaking2Me', which aims to capture social information without the need for all social contacts to use the technology. This system captures social information by using the audio signals only, which can operate on a single smartphone device only. The system makes use of a deep learning autoencoder to extract encoded representations of people's voices and then calculates speaker identification and speaker counting using a trained binary classification neural network model.

This system is designed to minimise memory footprint of the trained model, and does not require the training of voice samples from participants social contacts.

These systems have no reliance on the existing infrastructure of buildings in which conversations may occur, and can work anywhere. This is an advantage over many of the systems presented in the literature. Furthermore, there is also no need to train these systems with new data from the social contacts of participants. These are both social sensing solutions, which are scalable and are easy to deploy.

This thesis also explored two main research questions: (1) how can smartphones accurately track social interactions in different scenarios and (2) how can a suitable sensing system be scalable and how can it be utilised to capture the social context from other participants who are not using the sensing system. The work in Next2Me primarily explored the tracking of social interactions between human participants in different social scenarios, and the key findings were that we can accurately detect these scenarios with 88% accuracy in challenging scenarios, where different social groups are within close proximity but actually contain unrelated clusters of interacting people. Speaking2Me, however, looked into the scalability issues that can occur with sensing systems, such as all participants needing to install a tracking app. A key finding here, was that only one user can need to install the sensing app, and by utilising unsupervised deep learning techniques, we can form a database of the social contacts that a participant interacts with on a daily basis.

5.1 Interpretation of the Findings

The work in this thesis hoped to track social interactions using smartphones and cloud processing as the primary tools. Generally speaking, continuous sensing when using smartphones is one of the biggest challenges. And without a comprehensive data collection, it would be much more difficult to make accurate interpretations of the data. In this section, we discuss the findings and the steps we took to sense social interactions, how we built towards a continuous sensing system and how we optimised the parameters needed to improve classification performance.

5.1.1 Detecting Social Interactions

Social interactions occur all the time in the real world. Ideally, a social sensing system would be able to capture information about it's surroundings, including what social interactions are occurring. In our work, we proposed to use WiFi to capture these, due to the vast availability of 2.4 GHz WiFi in urban areas. However, ultimately there were limitations of using WiFi signals to capture social interactions, for example, if we consider a queue of people in a coffee shop, it is likely that most people are not part of the same social interaction and are instead queuing as individuals. A social sensing system is limited by what it can capture in these scenarios, primarily because the system would consider people who are co-located for a significant amount of time. Considering these limitations of using WiFi signals alone to detect social interactions, in Next2Me we explored the feasibility of also use the capture of audio signals as a way of distinguishing social groups that are in close proximity to each other.

We explored if sound signals can reveal distinctive patterns that can help differentiate between people participating in the same conversation, and developed a technique of using the top n frequencies in a speech sample to generate *"sound fingerprints"*. Based on the findings of preliminary studies, we designed a system that would detect social interactions using a combination of WiFi signals, as an early indicator that users are in close proximity, followed by audio sensing to identify smaller groups within the same area. Now, if we consider the same coffee shop queuing scenario, the hypothesis here is that the pattern of sounds between social groups would be different. For example, if you are queuing with a friend, perhaps you are facing towards each other and your smartphones are able to captures the speech signals based on this orientation setup. Other people in the queue, if interacting, would be speaking in different directions where human bodies would absorb the sound signals, and therefore the sound patterns from the other people would be different from your social group.

To test the system, we orchestrated various social scenarios where interacting groups are close but part of separate interactions. We used the WiFi data to detect when people are co-located (using similarity techniques for comparing the WiFi RSSI signals) and used this as a trigger point to capture audio from the interactions.

From the audio data captured, we extracted the top frequencies and fed these into a community detection algorithm to extract communities from he data by applying the Louvain community detection algorithm. The output of the community detection represents the output of the system; identifying the different groups interacting within close proximity of each other.

We also showed the Speaking2Me system, which worked by extracting encoded representations of people's voices and classifies them using a trained binary classifier. Speaking2Me aimed to allow the detection of social interactions using one smartphone only, targeting scenarios where metrics are needed to be obtained in real time about the socialness of a single-user. In order to do this, the system employed state-of-the art Deep learning techniques for speaker identification, to establish if a given speech sample recorded by a smartphone device belongs to a new or known speaker within a user's database of social contacts. By tracking the unique speakers contained within a historical database of voices, Speaking2Me can determine the count of speakers involved and match them with previous encounters. By doing this it can be used to create new ways of tracking personal socialness over time. In this work, the trained autoencoder model is small in size, and therefore good for mobile computing purposes. The binary classifier model is also small, around 4mb in size, and can easily be stored on a user's smartphone device or downloaded across a stream via cloud computing. We tested the performance of the this system through multiple auto-generated conversations to achieve average precision of 74.72%.

5.1.2 Conserving Energy Consumption

The design of the a social sensing system relies heavily on sensing modalities that can have a significant impact on the battery life of the participants' smartphones. In this thesis, we analysed the energy cost implications of using Next2Me. In our analysis we attempted to establish the average cost of using Next2Me in the form of electric charge (measured in mAh) consumed during a typical day. The aim of this estimation was to allow us to consider the impact that the system would have on the battery life of common smartphones, with battery capacities in the rage of 2,800mAh (Samsung S5) to 3,220mAh (Nexus 6). We also anticipated that precision on social interaction detection using sound fingerprints could be improved by combining multiple sound

fingerprints captured over longer periods of time, primarily because there were cases where a randomly selected 10sec sound fingerprint captured a situation where the actual social groups are not correctly mapped, perhaps when speech was not present from the social group or from moments where background noise was too large. The main hypothesis here was to (i) capture fingerprints of audio using duty cycling to reduce energy consumption and (ii) to combine fingerprints these fingerprints by calculating an average over n samples to improve the accuracy.

We experimented with combining multiple fingerprints for the detection of social groups. This involved making modifications in the way that the weights in the social network graph were calculated. Specifically, when the social graphs were formed, the weight between two connected nodes was calculated using the average of the fingerprint similarity over the number of n sound fingerprints recorded with a duty cycling gap. In our experiments, we looked at combining 1, 2 and 3 fingerprints, recorded at 20 seconds, 40 seconds and 60 seconds apart (sleeping). The results showed that combining more fingerprints produced better precision, across multiple experimental setups. Specifically, a duty cycling scheme of 3 sound fingerprints with 40sec sleeping shows an average precision of 92% and a combination of 2 sound fingerprints with 40sec shows an average precision of 89%. Following this, we concluded that for a setup of 3 samples/40sec sleeping is appropriate for the higher precision, whilst a 2 samples/40sec sleeping scheme offers a good balance of energy cost and precision. We chose to use 2 samples/40sec sleeping scheme, and this boosted the performance of the system by 9.8% when compared to a system which is truly continuously sensing. Moreover, the power consumption is reduced, due to the system no longer needing to sense all of the time.

5.1.3 Optimisation

Speaking2Me introduced the work which used a Deep Learning Autoencoder to learn encoded representations of a user's voice. When using a Deep learning model, the parameters need to be tuned in order to achieve the optimum results. Optimising these hyperparameters can be a time consuming task, particularly if the process is done manually where new configurations are done by hand. Therefore, we used the Talos library [153] to carry out the configuration performance automatically via

optimisation algorithms such as Random Search, Grid Search, and correlation-based optimisation through an implementation written in Python and designed to work with TensorFlow/Keras directly. The idea here was to automate the testing of multiple permutations of hyperparameters to ensure that we are optimising the model to achieve the best classification result. This was split into two processes:

- An initial pass, for determining how parameters respond well when tuned
- A refined pass, for refining each parameters value to be most optimal

For the configuration, we chose these specific parameters to investigate:

1. **Learning rate** - to investigate the effect this has on the validation loss over multiple permutations
2. **Batch size** - to investigate which batch size performs the best
3. **Layer activation** - to investigate the effect of different layer activation techniques
4. **Optimizer technique** - to determine which method from the Keras library performs best
5. **Filter Size** - to determine the best number of filters (n) for layer one and the corresponding number of filters ($n * 2$) for layer two

5.1.3.1 Initial Pass

We ran Hyperparameter Optimization for over 500 permutations to produce a collection of different model architectures and their validation loss. This was done to develop an understanding of which parameters would best affect the validation loss during training. For each permutation, the validation loss was calculated as the point where *Early Stopping* had terminated after 3 consecutive epochs. We compared the hyper-parameters to produce graphs which would visualise any obvious trends in parameter selection, so that a more refined pass (see: 5.1.3.2) could be ran. At this stage, a good batch size was unknown and this parameter was used to experiment with how specific parameters performed across a variable batch size. The parameters

we chose to investigate are available in table 5.1.

Parameter Group	**Value**
First Layer Activation	ReLU [149]
	PReLU [154]
	LeakyReLU [155]
	eLU [156]
	Tanh [157]
Second Layer Activation	ReLU
	PReLU
	LeakyReLU
	eLU
	Tanh
Optimizer	Nadam [158]
	Adam [159]
	RMSProp [160]
	Stochastic Gradient Descent (SGD) [161]
	Adagrad [162]
Learning Rate	1.00^{-2}
	1.00^{-3}
	1.00^{-4}
Batch Size	Random range from 8 to 128
Filters	Random range from 8 to 64

Table 5.1: Outline of the initial parameter boundaries for the initial Hyperparameter Optimization pass

Learning Rate

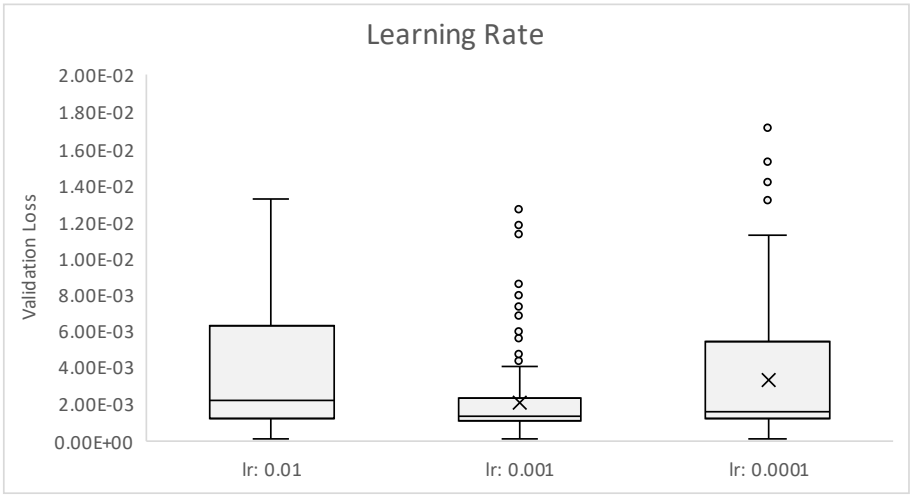


Figure 5.1: Box diagram to demonstrate the effect that learning rate has on the preliminary autoencoder model. lr is the "learning rate". We can see here that lr: 0.001 performs best because it has the lowest validation loss.

In figure 5.1 we can clearly see that a *learning rate* of 1.00^{-3} generally performs better than 1.00^{-2} and 1.00^{-4}. Based on these results, we decided that our refined pass at Hyperparameter Optimization would need to explore how the learning rate performs when using the values ranging between 1.00^{-3} and 1.00^{-4}, specifically values which incremented at 0.001.

Layer 1 Activation

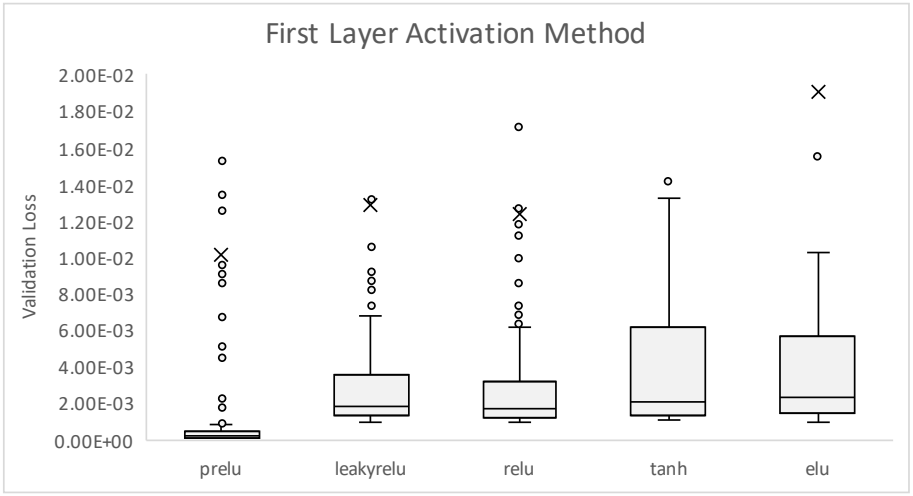


Figure 5.2: Box diagram to demonstrate the effect that the activation has on the first convolutional layer in the preliminary autoencoder model

In our model we applied a random activation function to the first layer to determine

which activation technique would perform best for our data set. We can see here in figure 5.2 that PReLU out-performed any other activation function, and so it was easy to conclude that in the refined pass, the first layer would need to always use PReLU activation.

Layer 2 Activation

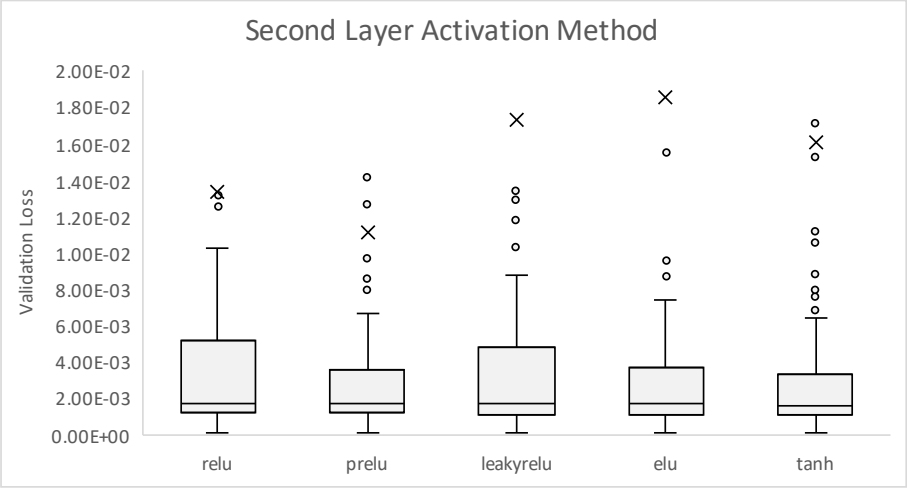


Figure 5.3: Box diagram to demonstrate the effect that the activation has on the second convolutional layer in the preliminary autoencoder model

For the second convolution layer (see figure 5.3), the activation method didn't seem to have a massive effect on reducing validation loss across the multiple permutations produced. Perhaps this was because the results of Hyperparameter Optimization included models where the first layer was not PReLU activation. However, for the final pass we considered all types of activation for the second layer for the refined pass, with the hopes of determining which activation function would work best with PReLU activated values obtained from the first layer.

Optimizer

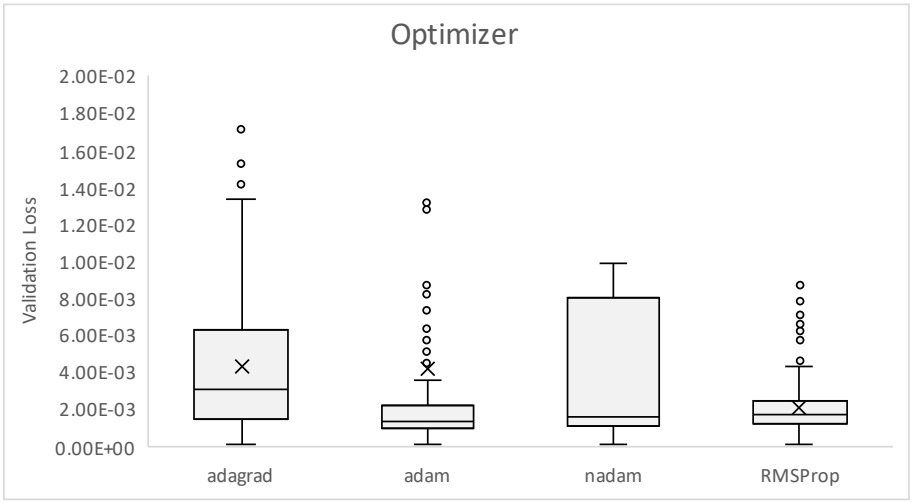


Figure 5.4: Box diagram to demonstrate the effect that the choice of optimizer has on the preliminary autoencoder model

Regarding the use of optimizers, we sorted the entire output of the initial pass by validation loss to discover that Nadam and Adam optimizers covered most of the top 10 of models produced. However, the above box chart in figure 5.4 shows that Adam performs better, whereas Nadam would typically under-perform across multiple permutations. Therefore, in conclusion from the results shown in figure 5.4, for future optimization it will be best to exclude Adagrad, SGD and Nadam optimizers, to focus more on how Adam and RMSProp optimizers perform.

5.1.3.2 Refined Pass

Using the results from section 5.1.3.1, we ran the Hyperparameter Optimization again as a *refined pass*. We used this to determine the final model for our speaker identification approach. The goal was to choose the best performing model out of numerous permutations whilst compromising validation loss to ensure that fewer learnable parameters were used, so that data size and computational cost would be lower and more plausible for use in a smartphone device. We ran the Hyperparameter Optimization again with a some modifications to capture the best model in a more rigorous fashion: the first layer only used PReLU activation, the optimiser was a selection between Adam and RMSProp, the learning rate would be a value between 1.00^{-4} and 1.00^{-3}, the batch size would be a number between 8 and 128, and the number of filters could range between 8 and 64 in multiples of 2. The number of

classes (40) remained the same. Table 5.2 outlines this setup. We ran hyperparameter optimization for 5750 permutations and recorded the results. Interestingly, the following charts have been scaled to a maximum of 0.0005 validation loss, from the original 0.02 validation loss, due to a significant initial improvement:

Parameter Group	**Value**
First Layer Activation	PReLU
Second Layer Activation	ReLU
	PReLU
	LeakyReLU
	eLU
	Tanh
Optimizer	Adam
	RMSProp
Learning Rate	Range from 1.00^{-4} to 1.00^{-3}
Batch Size	Random range from 8 to 128
Filters	Random range from 8 to 64

Table 5.2: Outline of the initial parameter boundaries for the refined Hyperparameter Optimization pass

Optimizer

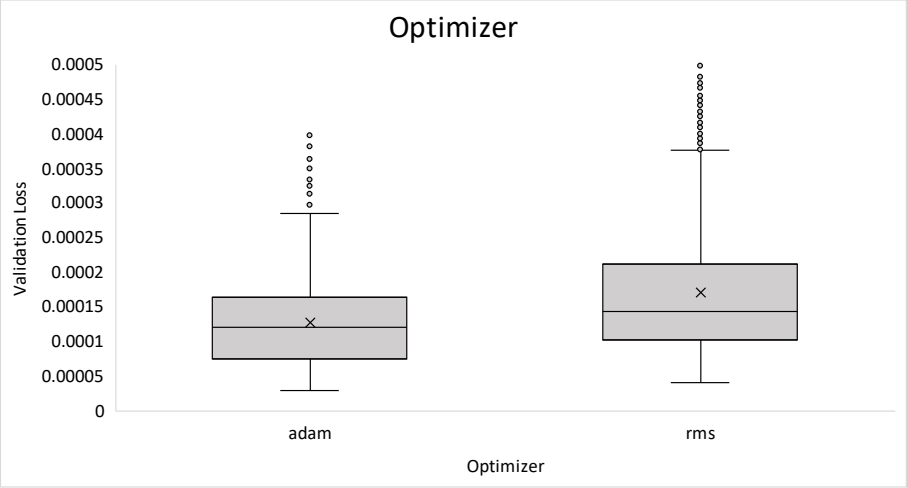


Figure 5.5: The effect that the optimizer choice (Adam or RMSProp) has on the validation loss of the model, for the refined pass of hyperparameter optimization

For the refined pass for deciding the best optimizer, it was a choice between Adam or RMSProp otimization techniques. Figure 5.5 shows that although the two perform quite well, Adam optimizer generally performs better. Due to this, Adam will be chosen as the optimization method in the final model architecture.

Batch Size

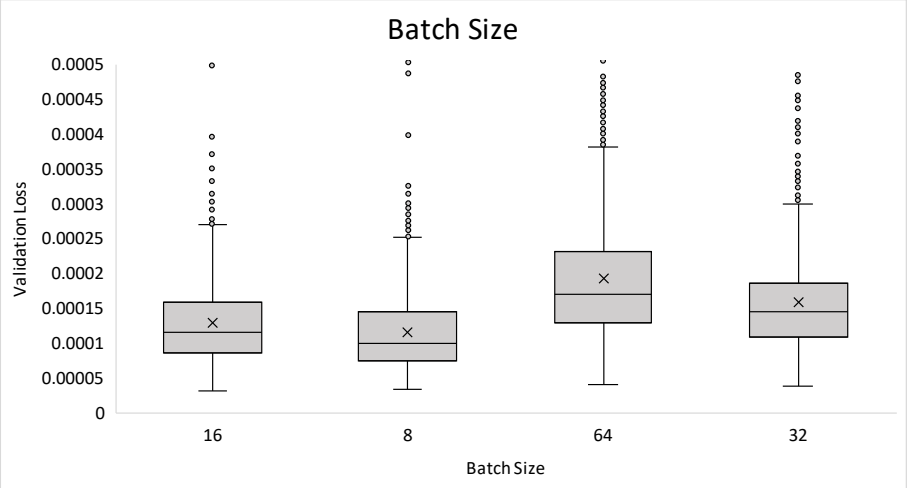


Figure 5.6: The effect batch size choice has on the validation loss of the model, for the refined pass of hyperparameter optimization

Figure 5.6 shows that in this architecture, a smaller batch size typically performs better. Therefore, a batch size of 8 was chosen for the final model.

Number of Filters

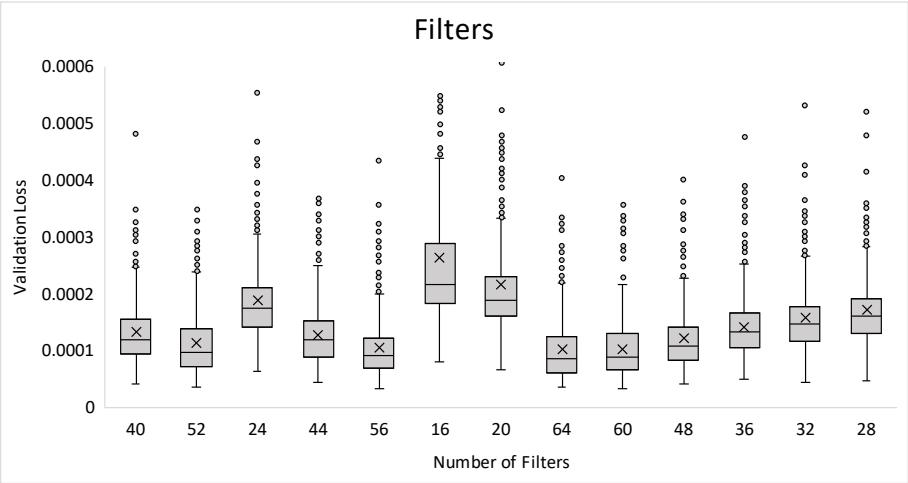


Figure 5.7: The effect that the number of filters has on the validation loss of the model, The effect that the number of filters has on the validation loss of the model, for the refined pass of hyperparameter optimization

Figure 5.7 shows that 40, 44, 48, 52, 56 and 64 filters will typically perform better than other filter sizes. Considering that the model should be as lightweight as possible, 64 filters will be used for the final model.

Learning Rate

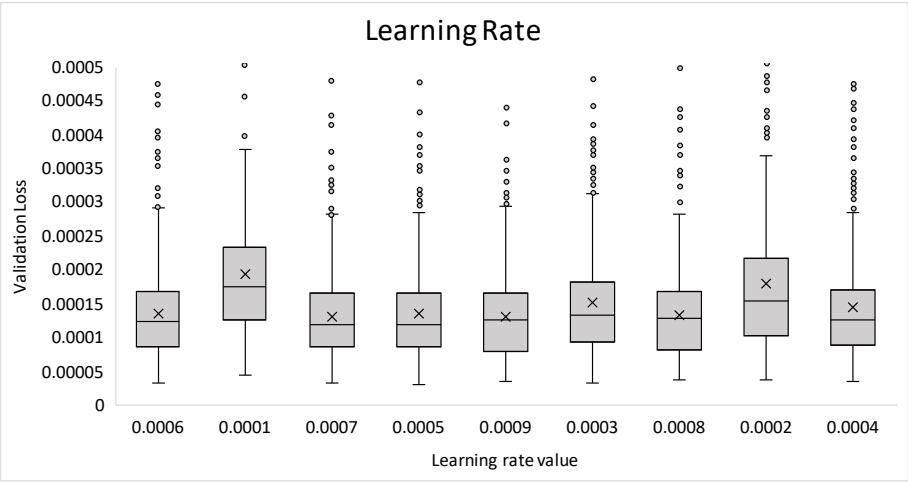


Figure 5.8: The effect that learning rate has on the validation loss of the model, for the refined pass of hyperparameter optimization. We can see here that most of the refined learning rate values have a similar validation loss

The figure in 5.8 shows that learning rate from 1.00^{-4} to 1.00^{-3} perform around the same. Therefore, we chose 5.00^{-4} as the learning rate for the final model, because it performed best.

Using the refined model presented in section 5.1.3.2, we trained the autoencoder model using all 1251 speakers from VoxCeleb (as described in section 4.6.1.1). We trained for 50 epochs, which was the moment that the Early Stopping detection determined that the validation loss could not be improved any more. The validation loss was calculated in the same was as in section 4.6.4. We also set the filters to 32 for the first layer and 64 for the second layer, set the optimiser to Adam, set the activation for the first layer to PReLU and PReLU for the second layer, set the learning rate to 5e-04 and set the batch size to 8. We validated against new audio files for the 1251 speakers, which consisted of speech recorded in different locations, with no overlap from the training data. We also validated each training and validation sample with a pair, as described in section 4.6.4.

5.2 Implications of the Findings

The results obtained from the experimentation in this thesis agreed with the literature and our original research questions. We wanted to look into how smartphones can accurately track social interactions in different scenarios and also a sensing system can be scalable and therefore utilised to capture the social context from other participants who are not using the sensing system. On Reflection, what we showed is that social interactions can be detected by using lightweight sensing modalities and classified without the use of Machine Learning. Moreover, we also showed that using state-of-the-art Deep Learning techniques, we can build a system which uses audio to build a speaker database of people who a user interacts with during social scenarios; by detecting new and existing speakers in an unsupervised way, we can count the number of meaningful social contacts across time. These results are ultimately different from other studies. For example, in Next2Me the work introduces "Sound Fingerprints" as a lightweight data transmission to the cloud, where a backend service can compare sound signals on demand and deliver contextual data to a frontend device (a smartphone) in real-time. This opens the door to new possibilities of tracking people, by being able to deliver the contextual information to the user in real-time. For example, if we consider a use-case where participants have memory issues, perhaps this system can be used to give the user instant information

about the social interaction such as participant names and if they have met the same person before.

In Speaking2Me we showed that by using state-of-the art Deep learning techniques for speaker identification, we can establish if a given speech sample recorded by a smartphone device belongs to a new or known speaker within a user's database of social contacts. The work in Speaking2Me mainly aimed to allow the detection of social interactions using one smartphone only, targeting scenarios where metrics are needed to be obtained in real time about the socialness of a single-user. By tracking the unique speakers contained within a historical database of voices, a social sensing system would be able to determine the count of speakers involved and match them with previous encounters, to create new ways of tracking personal socialness over time. Unlike most systems in the literature, Speaking2Me would work without the need to modify the existing infrastructure of a building, and without all participants needing to install the same app.

There are also, however, some practical implications for using these systems. There might be some issues with privacy - an issue which is repeatedly raised in the research topic. For Next2Me and Speaking2Me, it is unknown whether the top frequencies or encoded representations can be reconstructed to reveal spoken contents of a conversation. This may challenge the need to build a system which is robust against privacy concerns of the user.

In conclusion, the research conducted aims to help people. Whether people suffer from mental illnesses or whether people simply want to receive supplemental information about their day-to-day lives, these systems are a contribution to the collection of valuable digital data on the social scenarios that people experience.

5.3 Limitations and Future Work

In the design of Next2Me we focused on the detection of significant social interactions that last for more than 5 minutes. This technique is robust for such social events, but would not be appropriate for capturing short time and serendipitous interactions that last for only a few seconds. Although such short interactions are beyond the scope of

this work, the proposed technique could potentially be adapted with more aggressive use of sound sensing to capture such short events. However, such an approach would increase the power cost of sound sensing, and would require further exploration in adaptive sensing approaches to mitigate energy issues. Furthermore, the physical environment can have a significant impact on the performance of Next2Me.

In this work we demonstrated that the proposed system was robust against smartphone placement in participants' pockets, but further investigation would be necessary to fully explore the impact of the environment, such as higher/lower ceilings or significant acoustic echo. The implications of this limitation, is that reverberation could cause poor results, for example, in social scenarios where all participants are in a location where their voices echo or social scenarios with low ceilings, moving vehicles, etc. For future work, we will organise user studies where participants attend set up meetings and other social scenarios with a data collection app installed on their smartphone devices.

In Speaking2Me, we focused on resolving the limitations of Next2Me where all participants need to install the same smartphone app in order to curate an individual data set when they are involved in a social interaction. It's worth considering that the system would need to sense too aggressively, and this might lead to privacy concerns or power consumption concerns. However, the design of this system makes it feasible to assume that an interested individual can use a specialised wearable device that can track their social behaviour, or even utilise the technology within existing smartphone devices, however, using this approach with wearable devices would help to avoid the limitations imposed by mobile platforms and can help develop a more secure and privacy aware system. Furthermore, the opportunity to develop tailored power-saving solutions with hardware designed for audio sensing.

There is also future work that can be done within the exploration of the limitations of power saving approaches by using adaptive duty-cycling techniques; by exploring the additional context-aware triggers, we might be able to help implement an intelligent duty cycling system. For example, if we were able to listen to spatiotemporal triggers, we would be able to adapt duty cycling by discovering common patterns of social

interactions and the development of a probabilistic model where sensing is adapted according to the probability of social encounters in different contexts.

There is also room to explore the use of audio as a modality that can detect broader contexts beyond social interactions. Considering sound as a more generic sensing modality, it is possible to develop systems that can detect context such as the type of environment a person is in, or even the type of activities they perform. This is a broad area of researcher where in this work e.g audio fingerprinting and audio pattern embedding can be adjusted to capture more generic context.

CHAPTER 6

Conclusion

This dissertation explored the sensing of human social interactions using smartphone devices. We explored two main research questions: (1) how can smartphones accurately track social interactions in different scenarios and (2) how can a suitable sensing system be scalable and how can it be utilised to capture the social context from other participants who are not using the sensing system. We presented the experimentation and development of two main systems.

The first explored the tracking of social interactions between human participants in different social scenarios, and the key findings were that we can accurately detect social interactions from these scenarios with 88% accuracy in challenging scenarios, where different social groups are within close proximity but actually contain unrelated clusters of interacting people. This work also looked contributed to the issue of typical approaches considering the use of co-location as a proxy for real-world interactions and presented a collaborative system for capturing social interactions performed by social groups which are within close proximity but separate by nature by using WiFi data to determine if people are co-located and then compares the similarity of *audio fingerprints* to distinguish social groups in close proximity. Existing work can also requires special infrastructure and there is a lack of work that achieves good results in challenging scenarios such as co-located multiple social interactions

The second system, looked into the scalability issues that can occur with sensing

systems, such as all participants needing to install a tracking app. A key finding here was that only one user can need to install the sensing app, and by utilising unsupervised deep learning techniques, we can form a database of the social contacts that a participant interacts with on a daily basis. This work acted as a separate and scalable continuous tracking system which did not require all social contacts to use the same technology. This system can capture social information by using the audio signals only and uses a deep learning autoencoder to extract encoded representations of people's voices and then calculates speaker identification and speaker counting using a trained binary classification neural network model. Typical solutions that rely on audio, require training with voice samples of the participating users. All of these can cause issues with scalability and there was a clear need for the development of continuous social sensing systems that can operate in any environment.

6.1 Contributions

Our two systems contribute to solve the existing problems in the social sensing domain. Firstly, many systems require modifications to existing infrastructure, such as modification to WiFi access points or to install new technologies into the infrastructure itself. Our systems are both independent, and do not require any special infrastructure. The work in Next2Me only requires each participant to install an app onto their Android devices. The second system, Speaking2Me is a sensing system which can be turned into an Android app for a single participant to run on their device. Moreover, the models from Speaking2Me are small enough to be packaged inside the Android app itself.

Some systems require training data captured from newly enrolled speakers. Both of our systems do not require any re-training of the systems. The technique Speaking2Me is pre-trained and deployed as two standalone models: the autoencoder for producing encoded representations of people's voices, and a binary classification model.

Other systems are supervised and do not scale well when new speakers are introduced. Both of our proposed systems are unsupervised, and can therefore scale well when new social contacts are encountered. Next2Me can capture social interactions with 88%

precision and Speaking2Me can perform speaker identification with 75% precision.

6.2 Reflection

Reflecting on the original research questions, we wanted to look into how smartphones can accurately track social interactions in different scenarios. The work conducted in the Next2Me system answers this question by presenting it's novel and lightweight methodology for tracking human social interactions in diverse environments using Smartphone Devices and with the help of cloud computing. We also wanted to investigate how a sensing system can be scalable and how can it be utilised to capture the social context from other participants who are not using the sensing system. The work in the Speaking2Me answers this research question by making use of state-of-the-art deep learning methodologies to capture a list social contacts that a user might interact with on a daily basis. By automating the testing of this system, we show good precision of detecting a count of speakers involved in an interaction tested against 100 randomly generated conversations.

6.3 Future Work

These systems would make great applications for healthcare. For example, a system that can facilitate the monitoring of the quality of life for individuals, including social interactions. They might also be able to provide assistance for people with social anxiety or bipolar disorder to track their daily interactions and to bring awareness about how social they are being. This might be approached by use Speaking2Me system as a smartphone app to run experiments, which would use the count of speakers as an indication of social levels for participants, and then explore correlation between socialness and condition changes.

There might also be contributions to the transformation of solutions for custom made wearable devices. In Speaking2Me, the detection of social interactions can happen by a single individual. The design of this system makes it feasible to assume that an interested individual can use a specialised wearable device that can track their social behaviour. This approach would avoid the limitations imposed by mobile platforms

and can help develop a more secure and privacy aware system. Furthermore, the opportunity to develop tailored power-saving solutions with hardware designed for audio sensing.

The work carried out in Next2Me showed that continuous sensing is possible using lightweight sensing modalities. This can be used to develop and deploy large scale sensing systems, including social contact tracing, memory augmentation and more. However, these systems will never be used if the users do not trust them. More work needs to be done to explore privacy concerns. The literature touches on this topic quite a bit, and it would be my recommendation to explore privacy in social sensing systems. Without the trust of users, large scale data collection is simply not possible as an independent and optional app on a smartphone.

There is also future work to be done in the exploration of power saving approaches using adaptive duty-cycling. By exploring the additional context aware triggers, we might be able to help implement an intelligent duty cycling system, for example, listening to spatiotemporal triggers to adapt duty cycling by discovering common patterns of social interactions and the development of a probabilistic model where sensing is adapted according to the probability of social encounters in different contexts.

Moreover, the development of these systems should be continued. For example, in Next2Me more work can be done to test the system against social interactions which involve reverberation of restricted infrastructure (low ceilings, vehicles, corridors are good examples). Since it's original presentation, this work has also been cited numerous times [163–165]. It is my recommendation that the review and comments from others should be taken into consideration and can hopefully lead to the improvement of this work.

Furthermore, is also room to explore the use of audio as a modality that can detect broader contexts beyond social interactions. Considering sound as a more generic sensing modality, it is possible to develop systems that can detect context such as the type of environment a person is in, or even the type of activities they perform. This is a broad area of research where in this work e.g audio fingerprinting and audio

pattern embedding can be adjusted to capture more generic context.

Thank you for reading this thesis. I hope that my contributions will influence and inspire the future research into Social Sensing, particularly with using Smartphone Devices. And I also hope that the work outlined in this thesis will motivate the community to explore the use of audio as a rich sensing modality in ubiquitous computing to build really cool and useful systems that can help people.

References

[1] Sunmin Lee, Jinah Kim, and Nammee Moon. Random forest and wifi fingerprint-based indoor location recognition system using smart watch. *Human-centric Computing and Information Sciences*, 9(1):6, 2019.

[2] Beakcheol Jang and Hyunjung Kim. Indoor positioning technologies without offline fingerprinting map: A survey. *IEEE Communications Surveys & Tutorials*, 21(1):508–525, 2018.

[3] Lorenz Schauer. Analyzing the digital society by tracking mobile customer devices. In *Digital Marketplaces Unleashed*, pages 467–478. Springer, 2018.

[4] William Huang, Ye-Sheng Kuo, Pat Pannuto, and Prabal Dutta. Opo: a wearable sensor for capturing high-fidelity face-to-face interactions. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 61–75. ACM, 2014.

[5] Satyam P Todkar, Snehal S Babar, Rudrendra U Ambike, Prasad B Suryakar, and JR Prasad. Speaker recognition techniques: A review. In *2018 3rd International Conference for Convergence in Technology (I2CT)*, pages 1–5. IEEE, 2018.

[6] Khan Suhail Ahmad, Anil S Thosar, Jagannath H Nirmal, and Vinay S Pande. A unique approach in text independent speaker recognition using mfcc feature sets and probabilistic neural network. In *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, pages 1–6. IEEE, 2015.

[7] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[8] Rachit Shukla. Keywords extraction and sentiment analysis using automatic speech recognition. *arXiv preprint arXiv:2004.04099*, 2020.

[9] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, pages 373–382. Springer, 2017.

[10] P Misra and P Enge. Special issue on global positioning system. *Proceedings of the IEEE*, 87(1):3–15, 1999.

[11] Jeroen HM Bergmann, Patrick M Langdon, Ruth E Mayagoitia, and Newton Howard. Exploring the use of sensors to measure behavioral interactions: an experimental evaluation of using hand trajectories. *PloS one*, 9(2):e88080, 2014.

[12] Yuan Lai and Constantine E Kontokosta. Quantifying place: Analyzing the drivers of pedestrian activity in dense urban environments. *Landscape and Urban Planning*, 180:166–178, 2018.

[13] Cathal Gurrin, Alan F Smeaton, Aiden R Doherty, et al. Lifelogging: Personal big data. *Foundations and Trends® in information retrieval*, 8(1):1–125, 2014.

[14] Liane Colonna. Legal and regulatory challenges to utilizing lifelogging technologies for the frail and sick. *International Journal of Law and Information Technology*, 27(1):50–74, 2019.

[15] Snigdha Das, Soumyajit Chatterjee, Sandip Chakraborty, and Bivas Mitra. Meetsense: A lightweight framework for group identification using smartphones. *arXiv preprint arXiv:1804.05055*, 2018.

[16] Sebastian Feese, Michael Joseph Burscher, Klaus Jonas, and Gerhard Tröster. Sensing spatial and temporal coordination in teams using the smartphone. *Human-centric Computing and Information Sciences*, 4(1):15, 2014.

[17] Sophie Skach, Patrick GT Healey, and Rebecca Stewart. Talking through your arse: Sensing conversation with seat covers. In *CogSci*, 2017.

[18] Omer Berat Sezer, Erdogan Dogdu, and Ahmet Murat Ozbayoglu. Context-aware computing, learning, and big data in internet of things: a survey. *IEEE Internet of Things Journal*, 5(1):1–27, 2017.

[19] StatCounter Global Stats. Mobile operating system market share worldwide. *[Online]* *https://gs.statcounter.com/os-market-share/mobile/worldwide*, 2018.

[20] Zhenyong Zhang, Wei Ma, Mikko Topi Loikkanen, and Mark Kuhns. Duty-cycling microphone/sensor for acoustic analysis, September 5 2017. US Patent 9,756,420.

[21] Petko Georgiev, Nicholas D Lane, Cecilia Mascolo, and David Chu. Accelerating mobile audio sensing algorithms through on-chip gpu offloading. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 306–318. ACM, 2017.

[22] P. Georgiev, N. D. Lane, K. K. Rachuri, and C. Mascolo. Dsp.ear: Leveraging co-processor support for continuous audio sensing on smartphones. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 295–309. ACM, 2014.

[23] Ryan Daws and TechForge Media. Antutu's latest benchmark tests ai chip performance [online] `https://artificialintelligence-news.com/2019/01/30/antutu-benchmark-ai-chip-performance/`, Jan 2019.

[24] Zhiqiang Gong, Ping Zhong, and Weidong Hu. Diversity in machine learning. *CoRR*, abs/1807.01477, 2018.

[25] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[26] Renata Khasanova and Pascal Frossard. Graph-based isometry invariant representation learning. *arXiv preprint arXiv:1703.00356*, 2017.

[27] Jon Baker and Christos Efstratiou. Speaking2me: Unsupervised speaker identification for smartphones using deep learning. In *Second UK Mobile, Wearable and Ubiquitous Systems Research Symposium*. MobiUK, 2019.

[28] Jon Baker and Christos Efstratiou. Next2me: Capturing social interactions through smartphone devices using wifi and audio signals. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous 2017, pages 412–421, New York, NY, USA, 2017. ACM.

[29] Chiara Lunerti, Richard M Guest, Ramon Blanco-Gonzalo, Raul Sanchez-Reillo, and Jon Baker. Environmental effects on face recognition in smartphones. In *2017 International Carnahan Conference on Security Technology (ICCST)*, pages 1–6. IEEE, 2017.

[30] Chiara Lunerti, Richard Guest, Jon Baker, Pablo Fernandez-Lopez, and Raul Sanchez-Reillo. Sensing movement on smartphone devices to assess user interaction for face verification. In *2018 International Carnahan Conference on Security Technology (ICCST)*, pages 1–5. IEEE, 2018.

[31] Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, volume 2, pages 775–784. Ieee, 2000.

[32] Pratik Palaskar OnkarPathak, Rajesh Palkar, and Mayur Tawari. Wi-fi indoor positioning system based on rssi measurements from wi-fi access points–a trilateration approach. *Int. J. Sci. Eng. Res*, 5(4):1234–1238, 2014.

[33] Mostafa Uddin and Tamer Nadeem. Spyloc: A light weight localization system for smartphones. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 72–80. IEEE, 2014.

[34] Geert Vanderhulst, Afra Mashhadi, Marzieh Dashti, and Fahim Kawsar. Detecting human encounters from wifi radio signals. In *Proceedings of the 14th*

International Conference on Mobile and Ubiquitous Multimedia, pages 97–108. ACM, 2015.

[35] Wei Wang, Jiayu Chen, Tianzhen Hong, and Na Zhu. Occupancy prediction through markov based feedback recurrent neural network (m-frnn) algorithm with wifi probe technology. *Building and Environment*, 138:160–170, 2018.

[36] Venet Osmani, Iacopo Carreras, Aleksandar Matic, and Piret Saar. An analysis of distance estimation to detect proximity in social interactions. *Journal of Ambient Intelligence and Humanized Computing*, 5(3):297–306, 2014.

[37] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner. Virtual compass: relative positioning to sense mobile social interactions. *Pervasive computing*, pages 1–21, 2010.

[38] Phongsak Prasithsangaree, Prashant Krishnamurthy, and Panos Chrysanthis. On indoor position location with wireless lans. In *The 13th IEEE international symposium on personal, indoor and mobile radio communications*, volume 2, pages 720–724. IEEE, 2002.

[39] Rong-Hong Jan and Yung Rong Lee. An indoor geolocation system for wireless lans. In *2003 International Conference on Parallel Processing Workshops, 2003. Proceedings.*, pages 29–34. IEEE, 2003.

[40] Huiyu Liu, Yunzhou Zhang, Xiaolin Su, Xintong Li, and Ning Xu. Mobile localization based on received signal strength and pearson's correlation coefficient. *International Journal of Distributed Sensor Networks*, 11(8):157046, 2015.

[41] G Wolfle, René Wahl, Philipp Wertz, Pascal Wildbolz, and Friedrich Landstorfer. Deterministic propagation model for the planning of hybrid urban and indoor scenarios. In *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2005.

[42] John C Stein. Indoor radio wlan performance part ii: Range performance in a dense office environment. *Intersil Corporation*, 2401, 1998.

[43] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[44] Lei Zhang, Yanjun Hu, Yafeng Liu, Jiaxiang Li, and Enjie Ding. Wicloc: A novel csi-based fingerprint localization system. *International Journal of Pattern Recognition and Artificial Intelligence*, 2017.

[45] Seyed Mohammad Nekooei and MT Manzuri-Shalmani. Location finding in wireless sensor network based on soft computing methods. In *2011 International conference on control, automation and systems engineering (CASE)*, pages 1–5. IEEE, 2011.

[46] Haibo Ye, Tao Gu, Xiaorui Zhu, Jinwei Xu, Xianping Tao, Jian Lu, and Ning Jin. Ftrack: Infrastructure-free floor localization via mobile phone sensing. In *2012 IEEE International Conference on Pervasive Computing and Communications*, pages 2–10. IEEE, 2012.

[47] Chenshu Wu, Zheng Yang, Yunhao Liu, and Wei Xi. Will: Wireless indoor localization without site survey. *IEEE Transactions on Parallel and Distributed Systems*, 24(4):839–848, 2013.

[48] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 261–272. ACM, 2009.

[49] Arsham Farshad, Jiwei Li, Mahesh K Marina, and Francisco J Garcia. A microscopic look at wifi fingerprinting for indoor mobile phone localization in diverse environments. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–10. IEEE, 2013.

[50] Xuke Hu, Jianga Shang, Fuqiang Gu, and Qi Han. Improving wi-fi indoor positioning via ap sets similarity and semi-supervised affinity propagation

clustering. *International Journal of Distributed Sensor Networks*, 11(1):109642, 2015.

[51] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

[52] Roee Hay. fastboot oem vuln: Android bootloader vulnerabilities in vendor customizations. In *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[53] Jun Qi and Guo-Ping Liu. A robust high-accuracy ultrasound indoor positioning system based on a wireless sensor network. *Sensors*, 17(11):2554, 2017.

[54] Mike Hazas, Christian Kray, Hans Gellersen, Henoc Agbota, Gerd Kortuem, and Albert Krohn. A relative positioning system for co-located mobile devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 177–190. ACM, 2005.

[55] Faheem Zafari, Athanasios Gkelias, and Kin K Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, 2019.

[56] Masayuki Murata, Dragan Ahmetovic, Daisuke Sato, Hironobu Takagi, Kris M Kitani, and Chieko Asakawa. Smartphone-based localization for blind navigation in building-scale indoor environments. *Pervasive and Mobile Computing*, 57:14–32, 2019.

[57] Rohan Kumar Yadav, Bimal Bhattarai, Hui-Seon Gang, and Jae-Young Pyun. Trusted k nearest bayesian estimation for indoor positioning system. *IEEE Access*, 7:51484–51498, 2019.

[58] Wilson Sakpere, Michael Adeyeye-Oshin, and Nhlanhla BW Mlitwa. A state-of-the-art survey of indoor positioning and navigation systems and technologies. *South African Computer Journal*, 29(3):145–197, 2017.

[59] Wenjie Hu, Guohong Cao, Srikanth V Krishanamurthy, and Prasant Mohapatra.

Mobility-assisted energy-aware user contact detection in mobile social networks. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 155–164. IEEE, 2013.

[60] Niklas Palaghias, Seyed Amir Hoseinitabatabaei, Michele Nati, Alexander Gluhak, and Klaus Moessner. Accurate detection of real-world social interactions with smartphones. In *2015 IEEE International Conference on Communications (ICC)*, pages 579–585. IEEE, 2015.

[61] Piotr Sapiezynski, Arkadiusz Stopczynski, David Kofoed Wind, Jure Leskovec, and Sune Lehmann. Inferring person-to-person proximity using wifi signals. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(2):24, 2017.

[62] Jenny Röbesaat, Peilin Zhang, Mohamed Abdelaal, and Oliver Theel. An improved ble indoor localization with kalman-based fusion: An experimental study. *Sensors*, 17(5):951, 2017.

[63] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[64] W. Bulten, A. C. V. Rossum, and W. F. G. Haselager. Human slam, indoor localisation of devices and users. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 211–222, April 2016.

[65] Marcos P Gerardo-Castro, Thierry Peynot, Fabio Ramos, and Robert Fitch. Robust multiple-sensing-modality data fusion using gaussian process implicit surfaces. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2014.

[66] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N Asokan, and Petteri Nurmi. Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentication. In *2014 IEEE International*

Conference on Pervasive Computing and Communications (PerCom), pages 163–171. IEEE, 2014.

[67] Ashish Kapoor, Rosalind W Picard, and Yuri Ivanov. Probabilistic combination of multiple modalities to detect interest. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 969–972. IEEE, 2004.

[68] Taemie Kim, Agnes Chang, Lindsey Holland, and Alex Sandy Pentland. Meeting mediator: enhancing group collaborationusing sociometric feedback. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 457–466. ACM, 2008.

[69] N. Palaghias, S. A. Hoseinitabatabaei, M. Nati, A. Gluhak, and K. Moessner. Accurate detection of real-world social interactions with smartphones. In *Communications (ICC), 2015 IEEE International Conference*, pages 579–585. IEEE, 2015.

[70] Aleksandar Matic, Venet Osmani, Alban Maxhuni, and Oscar Mayora. Multi-modal mobile sensing of social interactions. In *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 105–114. IEEE, 2012.

[71] Kleomenis Katevas, Katrin Hänsel, Richard Clegg, Ilias Leontiadis, Hamed Haddadi, and Laurissa Tokarchuk. Finding dory in the crowd: Detecting social interactions using multi-modal mobile sensing. *arXiv preprint arXiv:1809.00947*, 2018.

[72] S. A. Hoseinitabatabaei, A. Gluhak, and R. Tafazolli. udirect: A novel approach for pervasive observation of user direction with mobile phones. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference*, pages 74–83. IEEE, 2011.

[73] Timon Elmer, Krishna Chaitanya, Prateek Purwar, and Christoph Stadtfeld. The validity of rfid badges measuring face-to-face interactions. *Behavior research methods*, pages 1–19, 2019.

[74] Hossein Ahmadi, Nam Pham, Raghu Ganti, Tarek Abdelzaher, Suman Nath, and Jiawei Han. Privacy-aware regression modeling of participatory sensing data. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 99–112, 2010.

[75] Taiwoo Park, Jinwon Lee, Inseok Hwang, Chungkuk Yoo, Lama Nachman, and Junehwa Song. E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 260–273, 2011.

[76] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138, 2006.

[77] Yoni Halperin, Galit Buchs, Shachar Maidenbaum, Maya Amenou, and Amir Amedi. Social sensing: A wi-fi based social sense for perceiving the surrounding people. In *Proceedings of the 7th Augmented Human International Conference 2016*, pages 1–2, 2016.

[78] Long Jiang, Bin Gao, Jun Gu, Yuanpeng Chen, Zhao Gao, Xiaole Ma, Keith M Kendrick, and Wai Lok Woo. Wearable long-term social sensing for mental wellbeing. *IEEE Sensors Journal*, 19(19):8532–8542, 2018.

[79] Y. Lee, J. Song, C. Min, C. Hwang, J. Lee, I. Hwang, Y. Ju, C. Yoo, M. Moon, and U. Lee. Sociophone: Everyday face-to-face interaction monitoring platform using multi-phone sensor fusion. *Proceeding of the 11th annual international conference on Mobile systems, applications, and services - MobiSys '13*, 2013.

[80] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. Emotionsense: a mobile phones based adaptive platform for experimental social psychology research. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 281–290. ACM, 2010.

[81] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y. Chen, J. Li, and B. Firner. Crowd++: Unsupervised speaker count with smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 43–52. ACM, 2013.

[82] Simon Graf, Tobias Herbig, Markus Buck, and Gerhard Schmidt. Features for voice activity detection: a comparative analysis. *EURASIP Journal on Advances in Signal Processing*, 2015(1):91, 2015.

[83] Sophia Bano, Tamas Suveges, Jianguo Zhang, and Stephen J McKenna. Multimodal egocentric analysis of focused interactions. *IEEE Access*, 6:37493–37505, 2018.

[84] Martha Larson, Gareth JF Jones, et al. Spoken content retrieval: A survey of techniques and technologies. *Foundations and Trends® in Information Retrieval*, 5(4–5):235–422, 2012.

[85] Anil K Jain, Ruud Bolle, and Sharath Pankanti. *Biometrics: personal identification in networked society*, volume 479. Springer Science & Business Media, 2006.

[86] Shahab Pasha, Jacob Donley, and Christian Ritz. Blind speaker counting in highly reverberant environments by clustering coherence features. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1684–1687. IEEE, 2017.

[87] A. De Cheveigné and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[88] Namrata Dave. Feature extraction methods lpc, plp and mfcc in speech recognition. *International journal for advance research in engineering and technology*, 1(6):1–4, 2013.

[89] HC Romesburg. Cluster analysis for researchers (belmont, ca: Lifetime learning publications. *RomesburgCluster Analysis for Researchers1984*, page 149, 1984.

[90] Jorge Martinez, Hector Perez, Enrique Escamilla, and Masahisa Mabo Suzuki. Speaker recognition using mel frequency cepstral coefficients (mfcc) and vector quantization (vq) techniques. In *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, pages 248–251. IEEE, 2012.

[91] Anthony TC Goh. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9(3):143–151, 1995.

[92] James Lyons. Mel frequency cepstral coefficient (mfcc) tutorial. *Practical Cryptography*, 2015.

[93] Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang. Fully supervised speaker diarization. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6301–6305. IEEE, 2019.

[94] Jan Chorowski, Ron J Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *arXiv preprint arXiv:1901.08810*, 2019.

[95] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[96] Harishchandra Dubey, Abhijeet Sangwan, and John HL Hansen. Robust feature clustering for unsupervised speech activity detection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2726–2730. IEEE, 2018.

[97] John A Hartigan, Pamela M Hartigan, et al. The dip test of unimodality. *The annals of Statistics*, 13(1):70–84, 1985.

[98] Moo-Ryong Ra, Bodhi Priyantha, Aman Kansal, and Jie Liu. Improving energy efficiency of personal sensing applications with heterogeneous multi-processors.

In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1–10. ACM, 2012.

[99] Ozgur Yurur, Chi Harold Liu, Xue Liu, and Wilfrido Moreno. Adaptive sampling and duty cycling for smartphone accelerometer. In *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, pages 511–518. IEEE, 2013.

[100] D. Datta, P. P. Datta, and K. K. Majumdar. Role of social interaction on quality of life. *National Journal of Medical Research*, 5(4):290–292, 2015.

[101] J. E Perry-Smith and C. E Shalley. The social side of creativity: A static and dynamic social network perspective. *Academy of management review*, 28(1):89–106, 2003.

[102] M. R. Carillo, E. Papagni, and F. Capitanio. Effects of social interactions on scientists' productivity. *International Journal of Manpower*, 29(3):263–279, 2008.

[103] T. E Seeman. Social ties and health: The benefits of social integration. *Annals of epidemiology*, 6(5):442–451, 1996.

[104] S. Liu, Y. Jiang, and A. Striegel. Face-to-face proximity estimation using bluetooth on smartphones. *IEEE Transactions on Mobile Computing*, 13(4):811–823, 2014.

[105] A. Barrat, C. Cattuto, V. Colizza, J. Pinton, W. Van den Broeck, and A. Vespignani. High resolution dynamical mapping of social interactions with active rfid. *arXiv preprint arXiv:0811.4170*, 2008.

[106] C. Brown, C. Efstratiou, I. Leontiadis, D. Quercia, C. Mascolo, J. Scott, and P. Key. The architecture of innovation: Tracking face-to-face interactions with ubicomp technologies. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 811–822. ACM, 2014.

[107] X. Hu, J. Shang, F. Gu, and Q. Han. Improving wi-fi indoor positioning via ap

sets similarity and semi-supervised affinity propagation clustering. *International Journal of Distributed Sensor Networks*, 11(1):109642, 01/14; 2017/02 2015. doi: 10.1155/2015/109642; 28.

[108] M. B. Kjægaard and C. V. Munk. Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength (concise contribution). In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 110–116. IEEE, 2008.

[109] M. B. Kjærgaard, M. Wirz, D. Roggen, and G. Tröster. Mobile sensing of pedestrian flocks in indoor environments using wifi signals. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pages 95–102. IEEE, 2012.

[110] H. Hong, C. Luo, and M. C. Chan. Socialprobe: Understanding social interaction through passive wifi monitoring. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 94–103. ACM, 2016.

[111] G. Vanderhulst, A. Mashhadi, M. Dashti, and Fahim Kawsar. Detecting human encounters from wifi radio signals. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, pages 97–108. ACM, 2015.

[112] I. Rishabh, D. Kimber, and J. Adcock. Indoor localization using controlled ambient sounds. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference*, pages 1–10. IEEE, 2012.

[113] B. Thiel, K. Kloch, and P. Lukowicz. Sound-based proximity detection with mobile phones. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*, page 4. ACM, 2012.

[114] A. Clifford and J. Reiss. Proximity effect detection for directional microphones. In *Audio Engineering Society Convention 131*. Audio Engineering Society, 2011.

[115] Z. Liu, Z. Zhang, L. He, and P Chou. Energy-based sound source localization and gain normalization for ad hoc microphone arrays. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–761. IEEE, 2007.

[116] C Rader and NJIToA Brenner. A new principle for fast fourier transformation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(3):264–266, 1976.

[117] Johannes Abel and Tim Fingscheidt. Sinusoidal-based lowband synthesis for artificial speech bandwidth extension. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):765–776, 2019.

[118] Herbert L Pick Jr, Gerald M Siegel, Paul W Fox, Sharon R Garber, and Joseph K Kearney. Inhibiting the lombard effect. *The Journal of the Acoustical Society of America*, 85(2):894–900, 1989.

[119] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[120] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[121] R. Lambiotte, J. Delvenne, and M. Barahona. Laplacian dynamics and multiscale modular structure in networks. *arXiv preprint arXiv:0812.1770*, 2008.

[122] A. Rice and S. Hay. Decomposing power measurements for mobile devices. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pages 70–78. IEEE, 2010.

[123] Wifimanager issue - google groups, 2019.

[124] Patrick Lazik, Niranjini Rajagopal, Bruno Sinopoli, and Anthony Rowe. Ultrasonic time synchronization and ranging on smartphones. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 108–118. IEEE, 2015.

[125] David Mills. Network time protocol (version 3) specification, implementation and analysis. Technical report, University of Delawar, 1992.

[126] Su Ping. Delay measurement time synchronization for wireless sensor networks. *Intel Research Berkeley Lab*, 6:1–10, 2003.

[127] Sathiya Kumaran Mani, Ramakrishnan Durairajan, Paul Barford, and Joel Sommers. A system for clock synchronization in an internet of things. *arXiv preprint arXiv:1806.02474*, 2018.

[128] Jun Gu, Bin Gao, Yuanpeng Chen, Long Jiang, Zhao Gao, Xiaole Ma, Yong Ma, and Wai Lok Woo. Wearable social sensing and its application in anxiety assesment. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 305–308. IEEE, 2017.

[129] Till Beiwinkel, Sally Kindermann, Andreas Maier, Christopher Kerl, Jörn Moock, Guido Barbian, and Wulf Rössler. Using smartphones to monitor bipolar disorder symptoms: a pilot study. *JMIR mental health*, 3(1):e2, 2016.

[130] Hande Hong, Chengwen Luo, and Mun Choon Chan. Socialprobe: Understanding social interaction through passive wifi monitoring. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS 2016, pages 94–103, New York, NY, USA, 2016. ACM.

[131] Zixing Zhang, Bingwen Wu, and Björn Schuller. Attention-augmented end-to-end multi-task learning for emotion prediction from speech. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6705–6709. IEEE, 2019.

[132] Jaebok Kim, Gwenn Englebienne, Khiet P Truong, and Vanessa Evers. Towards speech emotion recognition" in the wild" using aggregated corpora and deep multi-task learning. *arXiv preprint arXiv:1708.03920*, 2017.

[133] Srinivas Parthasarathy and Carlos Busso. Jointly predicting arousal, valence

and dominance with multi-task learning. In *INTERSPEECH*, pages 1103–1107, 2017.

[134] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE, 2018.

[135] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Advances in neural information processing systems*, pages 4480–4490, 2018.

[136] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.

[137] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 41.1–41.12. BMVA Press, September 2015.

[138] Davis E King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009.

[139] Joon Son Chung and Andrew Zisserman. Out of time: automated lip sync in the wild. In *Asian conference on computer vision*, pages 251–263. Springer, 2016.

[140] Md Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech communication*, 54(4):543–565, 2012.

[141] HB Kekre, Vaishali Kulkarni, Prashant Gaikar, and Nishant Gupta. Speaker identification using spectrograms of varying frame sizes. *International Journal of Computer Applications*, 50(20), 2012.

[142] Vibha Tiwari. Mfcc and its applications in speaker recognition. *International journal on emerging technologies*, 1(1):19–22, 2010.

[143] Kshitiz Kumar, Chanwoo Kim, and Richard M Stern. Delta-spectral cepstral coefficients for robust speech recognition. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4784–4787. IEEE, 2011.

[144] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE, 2014.

[145] Yanick Lukic, Carlo Vogt, Oliver Dürr, and Thilo Stadelmann. Speaker identification and clustering using convolutional neural networks. In *2016 IEEE 26th international workshop on machine learning for signal processing (MLSP)*, pages 1–6. IEEE, 2016.

[146] Ondrej Novotny, Oldrich Plchot, Pavel Matejka, and Ondrej Glembek. On the use of dnn autoencoder for robust speaker recognition. *arXiv preprint arXiv:1811.02938*, 2018.

[147] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, volume 2013, pages 436–440, 2013.

[148] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[149] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[150] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[151] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[152] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[153] Mikko Kotila. Autonomio talos [computer software]. Retrieved from `http://github.com/autonomio/talos`, 2019. [Online; accessed July 2019].

[154] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[155] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[156] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[157] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[158] Timothy Dozat. Incorporating nesterov momentum into adam. *ICLR 2016 workshop*, 2016.

[159] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[160] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[161] Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of

a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[162] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[163] Snigdha Das, Soumyajit Chatterjee, Sandip Chakraborty, and Bivas Mitra. Groupsense: A lightweight framework for group identification. *IEEE Transactions on Mobile Computing*, 18(12):2856–2870, 2018.

[164] Snigdha Das. A framework for group identification using smartphone and wearables. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 1847–1850, 2018.

[165] Ting-Han Chen, Sok-Ian Sou, and Yinman Lee. Witrack: Human-to-human mobility relationship tracking in indoor environments based on spatio-temporal wireless signal strength. In *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 788–795. IEEE, 2019.

APPENDIX A

ETHICAL APPROVAL

Ethics Checklist "SocialSense" 0151617

Daiva Nacyte <D.Nacyte@kent.ac.uk>
Fri 09/12/2016 16:41

To: J.M.Baker <jb956@kent.ac.uk>
Cc: Andrew MacGregor <A.N.MacGregor@kent.ac.uk>; Jim Ang <C.S.Ang@kent.ac.uk>

Dear Jonathan

I am sorry if you have already received this message but I cannot find any record that it was sent to you so I am emailing you (again).

Thank you for completing and sending the Ethics Review Checklist for your research project entitled "SocialSense" to the Faculties Support Office.

As you have ticked "no" to all of the questions on the checklist, your project does not require ethical approval and you may proceed with your research. I will keep your checklist on file for our records.

Should you wish to make contact about this project in the future, please quote reference number 0151617.

Best wishes
Daiva

Miss Daiva Nacyte
Administrative Co-ordinator - Faculties Support Office University of Kent, Room 25, The Marlowe Building, Canterbury, CT2 7NR

T: 01227 82(4252) I W: http://www.kent.ac.uk/fso/

Figure A.1: The ethical approval application response received from the Faculties Support Office at the University of Kent - 09/12/2016

APPENDIX B

PARTICIPANT CONSENT FORM

Participant Consent Form
Next2Me: Detecting Social Interactions using Wi-Fi and Audio Fingerprinting in Smartphone Devices

Jon Baker, Dr Christos Efstratiou
School of Engineering & Digital Arts
University of Kent

I am a postgraduate research student in the School of Engineering and Digital Arts at the University of Kent. As part of my PhD thesis, I am conducting research under the supervision of Dr Christos Efstratiou and I am inviting you to participate in my study.

The purpose of the study is to use Android smartphone devices to try and detect when multiple people are involved in a meaningful social interaction. This methodology involves collecting nearby Wi-Fi access point data to discover if people are within the same location, and then using audio data to confirm if a social interaction is taking place.

This study involves installing an app onto your Android device to collect data about which Wi-Fi access points are discoverable by your device, as well as audio signals from the on-board microphone. This all happens inside an app called "Next2Me", which will automatically collect data, including: Wi-Fi data continuously (just the access point name and signal strength) and 10-second long audio recordings, only when you're near other participants.

The findings from this study will help with the development of algorithms that can increase the accuracy of detecting social interactions using Wi-Fi and audio signals captured by the device.

To participate in the study, you will be required to use an Android smartphone device, which may be provided with limited availability. The app should be installed onto your device, and kept running for the duration of the study. You will also need to enable location services for the duration of the study, since Android requires this for a Wi-Fi scan.

All data obtained in this study will be assigned using a unique identification number. Your name will not be associated to the identification number. Any data recorded from the smartphone device will be only used for research purposes and never shared. Your Wi-Fi and audio data will never be distributed and will be used purely for analysis purposes.

Your participation is completely voluntary. You may withdraw from this study at any time without penalty. You may also request for your data to be deleted at any time.

All data will be kept on a secure University server and disposed after no more than 12 months.

I confirm that I have read and understood the information above and have had the opportunity to ask questions. I understand that my participation is voluntary and that I am free to withdraw at any time, without giving any reason, and without penalty.

By signing this consent form, you are indicating that you fully understand the above information and agree to participate in this study.

Participant's signature: _____ Date: ____/____/____

Participant's name (please print): _____

If you have any further questions or concerns about this study, please contact Jon Baker at j.baker@kent.ac.uk

Figure B.1: Example of the participant consent form given to participants prior to enrolling in the Next2Me study