A Secure Lightweight Voice Authentication System (VAS) for IoT Environments

A Dissertation
Presented in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
with a
Major in Computer Science
in the
College of Graduate Studies
University of Idaho
by
Salahaldeen Duraibi

Major Professor: Frederick Sheldon, Ph.D.
Committee Members: Terence Soule, Ph.D.; Michael Haney, Ph.D.; Wasim Alhamdani, Ph.D.
Department Administrator: Terence Soule, Ph.D.

May 2021

Authorization to Submit Dissertation

This dissertation of Salahaldeen Duraibi, submitted for the degree of Doctor of Philosophy with a Major in Computer Science and titled "A Secure Lightweight Voice Authentication System for IoT Smart Device Users," has been reviewed in final form. Permission, as indicated by the signatures and dates below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor: _____ Date: _____
Frederick Sheldon, Ph.D.

Committee Members: _____ Date: _____
Terence Soule, Ph.D.

_____ Date: _____
Michael Haney, Ph.D.

_____ Date: _____
Wasim Alhamdani, Ph.D.

Department Administrator: _____ Date: _____
Terence Soule, Ph.D.

Abstract

The Internet of Things (IoT) provides everyday devices ways to identify and communicate with other devices and their users. The users can be human beings or other devices. The spectrum of IoT applications is very large and includes wearables such as body-area-network-devices and smartwatches. Other IoT application domains include smart homes, smart cities, e-health, etc. IoT devices have the capability for collecting information about surroundings, analyzing, and making decisions without user intervention. Therefore, security in IoT environments is a critical requirement. Particularly, authentication is of high interest given the potential damage that may result from unauthorized access to such devices. This dissertation proposes a biometric based authentication model that can be used in IoT devices. The survey of related work evaluated different biometrics used for authentication which were compared for their suitability in such an environment. In considering the resource-constrained nature of IoT devices, a voice biometrics is preferred compared to other biometrics, such as fingerprint, iris scan, keystrokes, among others that would need dedicated hardware devices in such ecosystems.

In the second part of the dissertation, two separate deep learning approaches that extract and classify voice features were studied to detect and prevent voice replay spoofing attacks. These approaches are based on a convolutional neural network (CNN) and a deep neural network (DNN). The CNN approach was used as a feature extractor, while the DNN approach was used as a classifier. In addition, hybrid features that consist of Mel frequency Cepstral Coefficients (MFCC) and Constant Q Cepstral Coefficients (CQCC) features were used to train both approaches. Compared to other state-of-the-art voice authentication approaches, both the CNN and DNN approaches improved the state of detecting spoof replay attacks in speaker verification. Finally, enticed by the results of the two approaches, a voice-based authentication system that can be used in a network of IoT devices was developed. Afterwards, the system was validated in an experiment using virtualized IoT devices. The authentication system has shown improved relative accuracy by 15.8 percent.

Acknowledgements

First and foremost, I want to thank GOD, the Almighty, for the completion of this dissertation. This work would not have been possible without the kind support and help of many individuals. I am extending my sincere thanks to all of them.

I would like to express my deepest gratitude to my Major Professor, Frederick T. Sheldon, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing this research. I am extremely grateful to the rest of my committee: Prof. Terence Soule, Prof. Michael Haney, and Prof. Wasim Alhamdani, for their encouragement and insightful comments.

Moreover, I am thankful to the Saudi Arabian Cultural Mission and Jazan University for funding my scholarship. Finally, I am deeply grateful to the University of Idaho's Computer Science faculty, staff and students for how they have positively affected my studies and for providing me with an environment that enabled me to work.

Dedication

I would like to dedicate this dissertation to praising GOD and asking for his peace and blessing on all of those who contributed to completing this research and making it useful knowledge. This work is also dedicated to:

My father, Nasser Duraibi,

My mother, Laila Althurwi,

My wife, Wejdan Althurwi,

My kids, Wassem and Wateen,

My brothers and sisters,

who have been a constant source of support and encouragement during the challenges of studying and life. I am truly thankful for having you in my life.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Chapter 1: Introduction

In the beginning of the last decade, the use of smart devices grew exponentially. These devices have a significant place in the everyday life of people. Moreover, people use their smart devices everywhere and at any time and consider them an important part of their lives. The Internet of Things (IoT) device users use applications on their smart phones for management and to control the IoT devices. People use remote access control mechanisms to communicate with smart devices at a distance from their homes. However, the verification of a user, before providing access to giving commands to devices, is an issue. For example, smart-home IoT device companies have concentrated their efforts on offering concrete novel authentication mechanisms for improved safety; this is because conventional authentication mechanisms, such as password or token-based authentications, fail to suit IoT technologies [1, 2]. Therefore, researchers have started investigating biometric-based authentication for IoT technologies [3, 4] that allow users to access their devices by face, iris, fingerprint, voice or other means of authentication; such biometric authentication is more user-friendly in nature compared to conventional authentication approaches. While such approaches can achieve higher accuracy in user authentication, they are subject to a variety of spoofing attacks [5, 6] and they also raise privacy concerns [7]. Speaker verification systems (SVSs) are vulnerable to spoofing attacks when an impersonator claims the identity of someone other than himself. Known spoofing attacks associated with SVSs include mimicry attacks, conversion attacks, and replay attacks. A mimicry attack is when a professional mimics the utterance of the voice of a genuine user to gain unauthorized access. Conversion attacks happen when voice transformation is performed on an imposter's utterance so that it sounds more like that of a genuine user. Finally, a replay attack is a spoof technique when pre-recorded speech is provided to speech verification systems, with the help of record-and-play devices to impersonate a genuine user. Among the three spoof attacks, replay attacks pose the most serious threat to SVSs.

Although advances have been made in the field of IoT authentication, more accurate and reliable authentication solutions are necessary that consider several issues introduced by the characteristics of IoT devices, which involve nagging pernicious security threats.

1.1 Preliminary Research Background

User authentication refers to the process in which a user submits his/her identity credential(s) (often represented by paired username and password) to an information system to validate that the person is who he/she claims to be. A biometric authentication system verifies the identity of a person based on either their unique physiological traits [8, 9] or their unique behavioral biometrics [3, 4]. Such a process remains a crucial concern with respect to an IoT-technologies context. However, the biometric authentication mechanisms proposed for IoT technologies concede several well-known drawbacks,

principally led by people's usage and behavior [5, 6]. In the literature, proposed authentication systems seem to have been designed without considering human behavior [10]. People do not use such authentication systems since they presume that using authentication systems may involve lots of trouble every time they want access to their smart devices [11]. For example, users have trouble remembering passwords; hence, they prefer to reuse the same password in multiple situations.

Moreover, most biometric authentication mechanisms proposed in the literature were designed for traditional desktop computers, where some require costly hardware or may require IoT devices to be powerful enough to run complex computations that such systems need to operate [12]. Accordingly, in this research, we point out two issues: proposed authentication schemes do not take into consideration users' needs and behaviors regarding their involvement in the authentication process; and, the importance of developing accurate, effective, and secure solutions that are able to run on resource-limited IoT devices without the need for an extra piece of hardware because the solutions are already adequately embedded.

1.2 Research Motivation and Objective

User authentication in the IoT ecosystem is problematic for some important reasons, including the resource-limitations of IoT devices, which makes conventional authentication schemes inefficient and unfeasible for IoT devices. In addition, the unattended nature of IoT devices has also made password-based authentication schemes much less useful for the IoT. Another main concern is that existing authentication schemes are easy to breach and hence cannot guarantee security. For example, if an attacker steals the credentials of a user to gain access to a smart door lock of a house or a health-monitoring smart device of a patient, this might be life threatening. In this light, security throughout the process of authentication must be maintained. Biometric authentication schemes are employed for IoT ecosystems instead of the commonly used password-based system for reasons of security and because they do not require users to remember different passwords for multiple devices. Nevertheless, such biometric-based approaches are prone to spoof attacks. One of the major bottlenecks for all these systems is that the user needs to interact actively with the system. One of the key aspects of IoT systems is that they are more human-centric in nature compared to the traditional network systems. Due to IoT's human-centric nature, the IoT system creates information about an individual's behavioral patterns that can be misused. Therefore, this research has attempted to answer the following questions:

1. Which kind of authentication mechanism is user-friendly and does not require additional hardware to be used with IoT devices?
2. What authentication feature will be difficult for the attacker who attempts to spoof the authentication systems?

3. How can the authentication scheme be deployed easily to demonstrate its usability?

1.3 Scope of the Research

The research was compelled by the following constraints:

- only considers cloud-connected IoT devices;
- develops a voice-based biometric authentication model (single-factor);
- only considers the replay spoofing attack to secure the authentication model;
- does not consider the fading nature of the voice of the speaker; and,
- uses pre-recorded "spoofed" (labeled) and "genuine" voices both in a pre-existing dataset.

1.4 Research Contribution

The main contribution of this dissertation is its novel machine learning authentication approach used and validated in the IoT context within the scope of research explained above. To this end, we first studied existing IoT authentication approaches from the literature and highlighted both their strengths and weaknesses. On this basis, we were able to suggest an authentication mechanism that is more usable, reliable and resilient to replay attacks. Our model, so called "Lightweight VAS" was developed that improves on the identified problems.

Subsequently, we presented authentication features used for detecting replay spoofing attacks that were realized in a practical implementation and assessed through rigorous experimentation. The experiments were conducted using an existing dataset that contains genuine and spoofed data. Consequently, we conducted several performance analyses and compared the results with the state-of-the-art computation performances in the literature. An improvement in the state-of-the-art performance was obtained, achieving 7.1 Equal Error Rate (EER), which compares to 11.3 EER in typical state-of-the-art systems.

1.5 Author's related publications

1. Salahaldeen Duraibi, Wasim Alhamdani and Frederick T. Sheldon "A Security Hardened End-to-End Lightweight IoT Voice Authentication System." Computers & Security; Elsevier, 2021 (Submitted at Elsevier)

2. Salahaldeen Duraibi, Wasim Alhamdani and Frederick T. Sheldon "Replay Spoof Attack Detection using Deep Neural Networks for Classification" 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA (Accepted at IEEE CSCI conference)

3. S. Duraibi, W. Alhamdani and F. T. Sheldon, "Voice Feature Learning using Convolutional Neural Networks Designed to Avoid Replay Attacks," 2020 IEEE Symposium Series on Computational

Intelligence (SSCI), Canberra, Australia, 2020, pp. 1845-1851, doi: 10.1109/SSCI47803.2020.9308489

4. Salahaldeen Duraibi, Fahad Alqahtani, Frederick T. Sheldon and Wasim Alhamdani "Suitability of Voice Recognition within the IoT Environment" Springer Nature publisher - Advances in Security, Networks, and Internet of Things; Security and Management research track (SAM 2020); in press 2020.

5. Salahaldeen Duraibi, Frederick T. Sheldon and Wasim Alhamdani. (2020) "Voice Biometric Identity Authentication Model for IoT Devices", International Journal of Security, Privacy and Trust Management (IJSPTM), Vol. 9, No.2.

6. Salahaldeen Duraibi, Abdullah Alashjaee and Jia Song. (2019) "A Survey of Symbolic Execution Tools", International Journal of Computer Science and Security (IJCSS)), Vol. 13, No.6.

1.6 Dissertation organization

The remainder of the dissertation is organized as follows: Chapter 2 discusses the research background; chapters 3 and 4 present deep learning approaches used for feature extraction and classification and their implementations in experiments. A novel authentication scheme that detects replay spoofing attacks is presented in Chapter 5; and, finally, Chapter 6 provides the conclusions and the suggestions for future works.

Chapter 2: Background and Related work

Behavioral biometric authentication is considered a promising approach for securing the Internet of Things (IoT) ecosystem. In this chapter, we investigated the need and suitability of employing voice recognition systems in the user authentication for the IoT devices. We first studied the current status of IoT authentication technologies and determined that existing authentication systems suffer from several deficiencies. Some systems need higher computing and energy resources to perform their work, which exacerbates the resource-constrained nature of IoT technology. Others suffer from security weaknesses. Tools and techniques used in accomplishing voice recognition systems were reviewed, and their appropriateness to the IoT environment are discussed. In the end, a voice recognition system is proposed for IoT user authentication. Subsequently, the usability of the proposed system model was investigated. Our implementation of the model shows that voice biometrics can be used remotely to authenticate the user on his/her IoT devices.

2.1 Introduction

In general, within the context of IoT, three factors of authentication are usually employed: something a user knows (e.g., a password); something a user has (e.g., a secure token); and something a user is (e.g., biometric characteristics). Passwords are the most common authentication mechanism. However, password and token-based authentications have many security issues [1, 2] and are not suitable for smart devices because of the unattended nature of IoT smart devices. A biometric authentication system verifies the identity of a person based on either their unique physiological traits [8, 9] or their unique behavioral biometrics [3, 4]. Biometric authentication is more user-friendly in nature than the approaches that rely on passwords and secure tokens. While physiological traits can achieve high accuracy in user authentication, they are subject to a variety of attacks [5, 6] and also raise privacy concerns [7]. Moreover, the accuracy of physiology-based mechanisms may be substantially degraded by environmental factors, such as the viewing angle, illumination, and background noise [13, 14]. In contrast, behavioral biometrics (i.e., key stroke, voice, or gait analysis) appear less sensitive to ambient light or noise [15, 16]. Using biometric authentication for securing the IoT ecosystem is a promising approach [1].

Due to the portability, stability, and privacy of the voice features, voice recognition authentication has attracted extensive attention and application in recent years [2]. Voice recognition systems are versatile, simple to use, and non-intrusive by nature. They are considered accurate and do not require specialized tools; just a smartphone is enough for remote authentication for various services. Likewise, among other biometric authentication parameters, voice is the *simplest* and *easiest* unimodal to require and use for user authentication [8]. As a result, in recent years, voice authentication has attracted various leading

technology companies. For example, Google has provided Android-based Trusted Voice to allow users to unlock their smartphones. Saypay's mPayment consumers use a voice password to conduct transactions [2]. In addition, Google has promoted the employment of automatic speaker recognition for authenticating users in the IoT [9]. Regarding the nature of the IoT ecosystem, especially its mobile remote control, the use of voice recognition for user authentication may produce an overwhelming advantage [3]. In addition, the IoT ecosystem-related advantages of voice biometrics include small storage, ease of transmission, and non-intrusiveness [4].

In this chapter, a voice recognition authentication system to be used in the IoT ecosystem is proposed. The resource limitation of the IoT devices and remote access are taken into consideration. For example, the systems use Mel-frequency Cepstral Coefficient (MFCC) to extract features and Support Vector Machines (SVMs) for user verification which are fundamental to Remote Speaker Identification [5]. Voice features that can be extracted from acquired voice data can be of high-level or low-level attributes. Low-level attributes, related to the vocal tract, are derived from spectral measurements, while the high-level attributes are derived from behavioral cues, such as dialect, word usage, conversation patterns, etc. High-level attributes are difficult to extract but are less sensitive to noise [6, 7]. For example, *Idiolectal* and *Prosody* identify [4] high-level attributes of a speaker's voice, while *Short-term spectral* identifies low-level attributes of the speech signals. The latter is the main source of individuality in speech [3]. In addition, low-level attributes are easy to extract and are most common when applied to user authentication systems. In this light, extraction of low-level cues is necessary for IoT user authentication.

2.2 Background

Interconnected environments such as Machine to Machine (M2M), Machine to Individual, or Individual to Individual are what make up the IoT ecosystem. In the IoT ecosystem smart objects can communicate among themselves, these things can detect each other, and everything can interact with each other and with the local environment. These interconnections are facilitated by remote sensing and tracking capabilities, and every entity is provided with data transfer through the internet via such protocols as Wi-Fi, ZigBee, or Bluetooth among others. In particular, organizations may need such data for business, social, or research analytics [13]. For this reason, a vast variety of information is stored, managed, and processed. Access to these private data needs to practice secure access control. Employing conventional authentication mechanisms, such as passwords is reported to have fallen short for the IoT ecosystem. Thus, biometric technology is considered a better substitute for the protection of IoT private data [14].

There are three processes in any voice authentication system. These are *Voice Feature Extraction*, *Speaker Modeling*, and *Decision Making*. The *feature extraction* process is where the voice signals are

converted into a sequence of frames. Each frame is a short window of the waveform with overlapping adjacent windows [17]. Considering the unique resource-constrained characteristics of IoT devices, our work focused only on *Short-term spectral qualities* [18], as this requires less computational resources. Additionally, the selection of appropriate feature extraction methods is crucial in this process because of how they influence the performance of the system. The two most popular spectral-feature extraction methods are *filter bank analysis* and *linear predictive coding* [7, 19, 20]. In *filter bank analysis*, the voice signals are pass through a bank of band-pass filters that cover a range of frequencies consistent with the transmission characteristics of the signal. The spacing of the filters can be either uniform or non-uniform, based on the perceptual criteria, such as the Linear Frequency Cepstral Coefficient [LFCC] or the MFCC; the latter provides a linear spacing [7, 21]. In the *linear predictive coding*, the speech signal can be modeled by a linear process prediction. Signals at each time step use unique and specific periods of preceding samples that capture the temporal evolution of the features from one speech segment to another [7, 21].

After features of the voice are extracted. Automatic Speaker Verification *(*ASV) has the ability to construct a model λ_s for each user where "s" is user and λ is the specific model. Such modeling depends on, for example, whether it is used for applications that use fixed words (text-dependent), or applications that use phonemes not seen in the enrollment data (text-independent). Speaker modeling methods can be of two categorizes, non-parametric or parametric. The non-parametric approaches include templates which are suitable for a text-dependent verification system [7]. The parametric speaker modeling includes *Vector Quantization* (VQ), *Gaussian Mixture Models (GMM), and Hidden Markov Models (HMM)*. In VQ a set of representative samples of the user's enrollment voice is constructed by clustering the feature vectors. *GMM* has been proven to be very effective in the cases of a text-independent speaker recognition [17], also referred to as a refinement of vector quantization. HMM is suitable for text-dependent ASVs and is used for access control of personal information or bank accounts. Among the three techniques, GMM has been proven very effective for phones, and may be the best candidate for IoT devices [22]. There are also some other non-parametric and parametric approaches in the literature. However, those presented in this chapter are the most common techniques implemented in ASV systems. The final process is decision making. In this process an 'accept' or 'reject' decision is delivered based on the verification models discussed above.

To measure the effectiveness of ASVs a number of parameters were studied. These parameters include the False Acceptance Rate (FAR) which is the number of attacks being incorrectly labelled as authentic by the system. False Rejection Rate (FRR) refers to the number of authentic interactions being

incorrectly rejected as attacks. Relative Operating Characteristics (ROC) represents a compromise between FAR and FRR because the ROC helps enable systems tp minimize both FAR and FRR [23].

2.3 Related Work

This section reviews different biometric authentication mechanisms employed in the IoT ecosystem. Generally, there are two main types of IoT authentication methods, including centralized and distributed architectures, as shown in Figure 2.1 and Figure 2.2, respectively. The centralized architecture uses a centralized server to manage the credentials used in the authentication, whereas in the distributed architecture, the authentication is accomplished point-to-point between the communicating parties [11]. Biometric authentication is achieved based on these two architectures. There are four basic biometric authentication performance-measuring strategies; they include accuracy, scale, security, and privacy. Elements such as enrollment, biometric reference, comparison, networking, and personal biometric criteria are common in biometric authentication systems. Biometric-based authentication systems use two other factors, which are physical and behavioral [11].

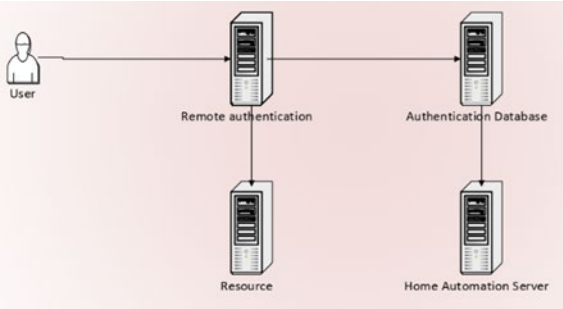

Figure 2. 1. Centralized authentication architecture

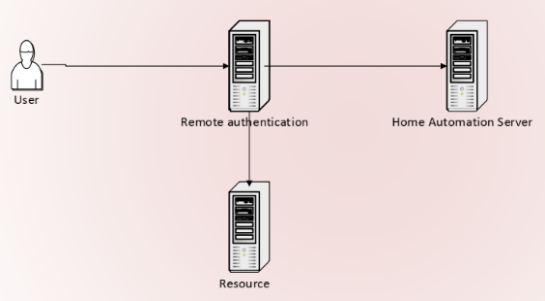

Figure 2. 2. Distributed authentication architecture

The physical factors include the recognition of fingerprint, face, iris, hand geometry, and palm print, while the behavioral factors may include, but are not limited to, voice, signature, and gait recognitions [14]. Biometric systems have some advantages over conventional identity-based methods (password and ID); they cannot be transferred, stolen, lost, broken, or easily guessed [15, 24]. The acceptance and

performance of the biometric systems are presented in Table 1. From Table 1 the voice biometric systems are more suitable compared to others biometric systems. The voice biometric systems use voice rather than complicated input methods, like fingerprints, that need special hardware for input. Therefore, voice is appropriate for the IoT where convenience is important. Some researchers have investigated and adopted voice-based biometric systems into the IoT ecosystem.

Table 2. 1 Comparison of biometric system

Factor Type	Biometric systems	Weakness	Strength
behavioural	Voice recognition	Has relatively low accuracy, inefficiencies in certain circumstances	Needs no hardware, ease of use, widespread usage, can be used for remote authentication
	Signature recognition	Has relatively low accuracy	Wide acceptance, non-rigging
	Detect Behaviour	Shows non-performance in certain conditions	Continuous authentication
	Tough dynamics	Inconsistent accuracy, lack of efficiency under certain conditions	Continuous authentication, does not require specific hardware
	Keystroke Dynamics	Inconsistent accuracy, lack of efficiency under certain conditions	Continuous authentication, does not require specific hardware
Physical	Fingerprint recognition	The need for additional hardware, the difficulty of obtaining high-quality images, the lack of efficiency in certain circumstances	Use and wide acceptance, low cost, good accuracy
	Face recognition	The need for additional hardware, lack of efficiency in certain circumstances	Use and wide acceptance, good accuracy and less fraud
	Iris recognition	The need for additional hardware, high cost, time-taking authentication	High precision, non-rigging
	Hand geometry recognition	The need for additional hardware, precision	Easy to use, less fraud
	Palm detection	The need for additional hardware, high cost	Public acceptance, high precision

For example, the researchers in [25] introduced a gaze feature-based model which is secure against iterative and side channel attacks. Likewise, in [26] the researchers used electrocardiogram for the development of their method in which they proved the good candidacy of biometric features for authentication of IoT devices. The results of the implementation of this scheme reveal that it has 1.41% FAR and 81.82% TAR for 4 seconds of signal acquisition. One of the main strengths of the scheme is that it conceals the biometric features during authentication, but the privacy preservation mechanism is not taken into consideration. The researchers argued in [27] for the suitability of signature-based authentication systems for the IoT device, whereby they presented three categories of a signature-based scheme, namely, offline, online, and behavior. Some Gait recognition based authentication systems proposed for IoT devices are also in the literature [28]. A touch screen based authentication scheme was proposed in [29].

In [30], a keystroke dynamics-based authentication scheme with three steps (enrollment, classifier and user authentication) was proposed. Similarly, in [31], a fingerprint-based authentication system was provided. In [32], the researchers introduced an authentication and authorization scheme that uses face recognition, which can be used for the IoT ecosystem. An iris-based authentication system used for unlocking mobile IoT devices was proposed in [33]. To the best of our knowledge, there are only two researchers who have adopted voice biometrics as an authentication mechanism for the IoT ecosystem. Shin and Jun [34] implemented voice recognition technology to verify authorized users for controlling and monitoring an automated home environment. The researchers proposed a voice recognition system that is divided into server and device parts. The role of the server part of the system is for user preregistration, user recognition, and control command analysis. The role of the device part is device command reception and device control then followed by response. The type of models and techniques employed is not discussed in this research. Likewise, the implementation of the model is not reported.

Oscar et al. [34] proposed a multimodal biometric approach for IoT, based on face and voice modalities. The researchers designed their system to scale to the limited resources of IoT technologies. For the voice recognition part of the system, Oscar et al. were able to extract MFCC features from the voice with the use of Fast Fourier Transform (FFT). Considering these findings, the filter banks are decorrelated with the application of a discrete cosine transform. However, this system does not fully utilize voice recognition. Although it has been implemented in a case study, the results cannot be compared to a system that fully utilizes voice recognition. Recall that the overall advantages of such biometric-based schemes are that they cannot be lost due to their intrinsic nature (e.g., fingerprints), they are very difficult to copy, they are hard to distribute without introducing artifacts into a copy, and they cannot be easily guessed. Conversely, conventional password-based authentication methods suffer from a number of drawbacks; they can be easily guessed, hacked, and/or cracked. The performance of the reviewed biometric systems is shown in Table 2.2. As shown in the table, only a few researchers have studied the areas related to the deployment of voice recognition systems applied to the IoT ecosystem for access control and user authentication. Building a working voice recognition system or integrating it to the IoT ecosystem is lacking in the literature. However, some projects conducted in the area of voice recognition in general do exist. Some are adapted to the mobile and cloud computing paradigms. The challenges of IoT devices' restricted resources of computation, storage, and power threaten the development of sophisticated authentication systems. Accordingly, a novel biometric approach has been proposed [14, 35].

Table 2. 2 Summary of Systems performance

Sources	Title	Method	FRR %	FAR %	ERR %
[1]	Multimodal Biometrics for Enhanced IoT Security	Voice & Face	81.62	N/A	8.04
[25]	Gazetouchpass: Multimodal authentication using gaze and touch on mobile devices	Gaze	N/A	N/A	0.32
[26]	ECG authentication for mobile devices	Keystroke	81.82	1.4	N/A
[27]	Behaviour based human authentication on touch screen devices using gestures and signatures	Signature	90	0	0.5
[28]	Edge-centric multimodal authentication system using encrypted biometric templates	Face	N/A	N/A	1.72%
[29]	Touchalystics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication	Touch dynamics	N/A	N/A	2-3%
[30]	Two novel biometric features in keystroke dynamics authentication systems for touch screen devices	Keystroke	8.40%	8.32%	8%
[31]	More efficient key-hash based fingerprint remote authentication scheme using mobile device	Fingerprint	N/A	N/A	N/A
[32]	Partial face detection for continuous authentication	Face	N/A	1%	N/A
[33]	Firme: Face and iris recognition for mobile engagement	Face & Iris	0.25	0.8	0.40%
[34]	Home IoT device certification through speaker recognition	Voice	N/A	N/A	N/A

2.4 Conventional Voice based biometric Authentication Model

A voice biometric-based IoT user authentication model is proposed as presented in Figure 2.3. The model has two phases: the enrollment phase when the user of the smart device says his voice for registration; and the verification phase when the system checks whether the identity claimer is the real user by comparing the previously enrolled voice with the voice of the identity claimer. Subsequently, if the similarity of the two voices reaches a certain predefined threshold, then access is granted to the claimer; otherwise the claimer is rejected. The design criteria of the model is given in Table 2.3. The following sections broadly discuss different components of the model.

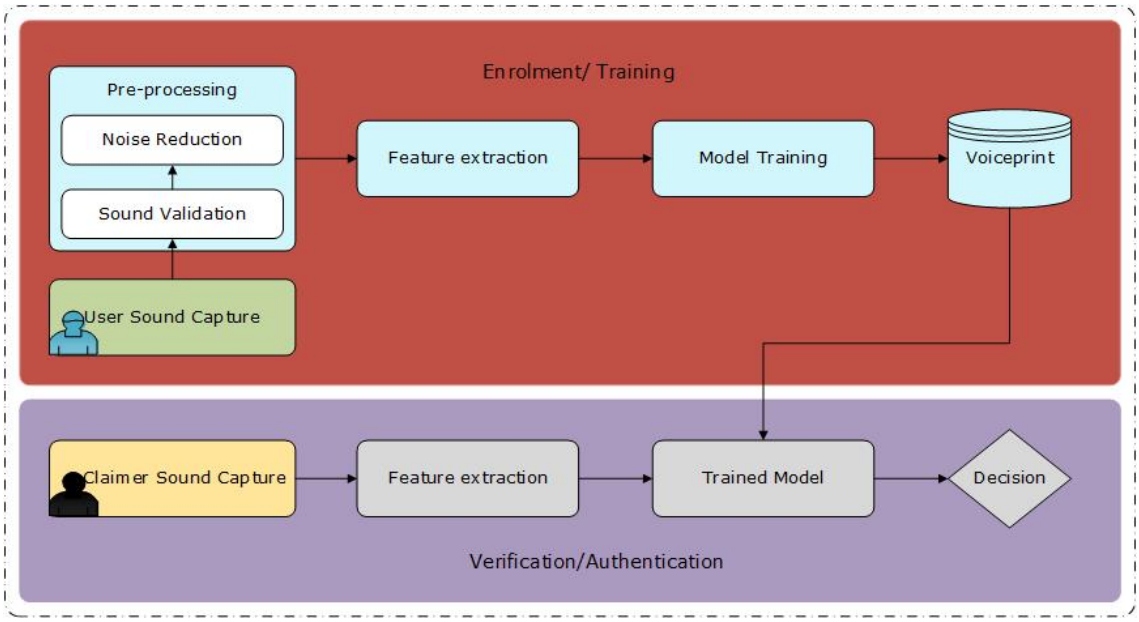

Figure 2. 3. High level IoT voice authentication model.

2.5 Enrollment Phase

This phase is where the user first introduces his voice for registration. Steps taken in this phase are discussed in the following subsection.

2.5.1 Sound capture

This step captures the sound or voice of the IoT device owner for training. This is expected to be done by the smartphone where the owner uses the control apps of the IoT devices. The output of this step is converted files with a suitable file format.

2.5.2 Pre-processing

In this step, the collected voice data is validated for defects. This is accomplished by decomposing the data at different frequencies at different scales, and the resulting wavelets are checked for the existence of any clipping. Subsequently, identified noise wavelets are removed and the noise-free data is

obtained. There are two known techniques for removing noise from collected sound data: with the use of the threshold based de-noising method [18, 36], or with the use of the recursive least-squares adaptive filtering method [37]. The first was adopted in our work for its suitability for smart devices.

Table 2. 3. Design criteria of the model

Design criteria	Description
Universality	A very high percentage of the population should have the characteristic. For example, virtually everyone has recognizable fingerprints, but there are rare exceptions.
Distinctiveness	No two people should have identical characteristics. For some otherwise acceptable characteristics, identical twins share virtually the same patterns, such as facial features and DNA, but not other features, such as fingerprints and iris patterns.
Permanence	The characteristic should not change with time. For otherwise acceptable characteristics, such as facial features and signatures, periodic re-enrollment of the individual may be required.
Collectability	Obtaining and measuring the biometric feature(s) should be easy, non-intrusive, reliable, and robust, as well as cost-effective for the application.
Performance	The system must meet a required level of accuracy, perform properly in the required range of environments, and be cost-effective.
Circumvention	The difficulty of circumventing the system must meet a required threshold. This is particularly important in an unattended environment, where it would be easier to use such countermeasures and a fingerprint prosthetic or a photograph of a face.
Acceptability	The system must have high acceptance among all classes of users. Systems that are uncomfortable to the user, appear threatening, require contact that raises hygienic issues, or are non-intuitive are unlikely to be acceptable to the general population

2.5.3 Feature extraction

Voice features deemed important for the system are extracted in this step. The extraction and selection of such feature vectors adds to the quality of the voice recognition system. Feature traits extracted from the owner's voice are expected to be different from that of others, must be robust related to noise and distortion, should be easily extractable, difficult for playback attacks, and should not change with the change of environment or health of the owner. As such, the most appropriate features used in this model are MFCC features. The MFCC coefficient was selected for its computing simplicity, which is suitable for the resource-constrained characteristics of the IoT ecosystem, as well as the mimicking nature of the human auditory capability of human ears. Likewise, the MFCC divides the voice signal into frames by subsequently applying a hamming window for every frame [34].

There are two well-known signal analysis tools that are used in existing voice recognition systems: Discrete Cosine Transformation (DST) and the Hidden Markov Model Toolkit (HTK). Hence, these tools take care of the details of obtaining the cepstral features of each frame.

2.5.4 Model Training

After extraction of the MFCC features, the voice model is trained for the IoT owner. The HMM model was adopted for this system; the reason is that HMMs are considered very effective for phones because the system app is used on a smartphone. Finally, the voiceprint is stored in database.

2.6 Verification/recognition Phase

Once the user enrollment phase is accomplished, the system is now expected to verify whether the identity claimer is the owner of the IoT device. The same steps of voice data collection and feature extraction are conducted on the claimer's sound via the smartphone. Subsequently, the extracted MFCC features are tested against the trained model for verification. SVMs are used in this step for training classifiers to provide a good generalization to automatically determine the verification data from the enrollment data. Finally, the decision is made for either rejection or acceptance. The authentication is rejected if the claimer's voice features fail to pass the test against the trained model.

2.7 Implementation of the Authentication Model

In this section, the proposed model is tested for its usability in the IoT ecosystem. The normal process of voice biometric implementation usually includes the creation of models from audio data, generation of tables, and self-authentication to the IoT manager. We, therefore, used a client-server model for the initial implementation. A user mobile device simulates running [1] a client, where the server simulates the IoT manager and handles the verification requests. By contrast, the IoT device receives the command from the IoT manager and automatically responds to the command. First, the user inputs his own voice command using the smartphone. Subsequently, the system on the server side determines whether the connection is for enrollment or verification and accordingly performs the process in each phase. New connections are considered first for enrollment, while returning connections are considered for verification by the server by prompting the identity claimer with challenge words.

2.7.1 Open Source Software

To accomplish the initial test, we used a software package called Mistral/Alize. Mistral is tested by the National Institute of Standards and Technology [NIST] and has with 0.5 error rate. Figure 2.4 shows the process of Mistral package.

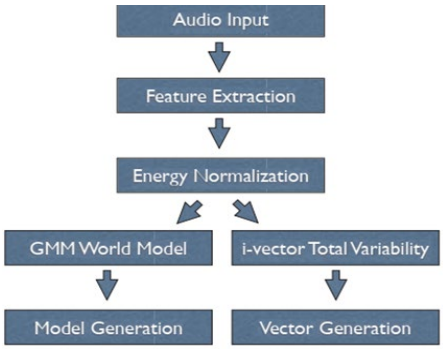

Figure 2. 4. The processing of the Mistral software package

2.7.2 Dataset

There are several datasets for audio dataset selection, but we selected the Massachusetts Institute of Technology (MIT) dataset, which appears to be recent and is considered to be good for initial results. MIT is designed for mobile environments. The dataset consists of 48 speakers including 22 female and 26 male. The dataset is used for the Universal Background Model (UBM) training.

2.7.3 Pre-processing

In this step, after the user inputs his voice using his own smartphone, the voice is validated, and noise is removed from the voice signal. For example, Figure 2.5 shows the raw voice signal without noise reduction, while Figure 2.6 illustrates that the noise (the silent part) is removed before it is submitted for feature extraction. In this process, using SPro [38], which is supported in the Mistral program, frame selection was performed by excluding silent frames longer than 100ms. In addition, if a sample is different from 16 KHz, SPro performs re-sampling by default. Mistral needs the voice to be more compact. The most important aspect of this process is that the voice is converted into the form of a binary format.

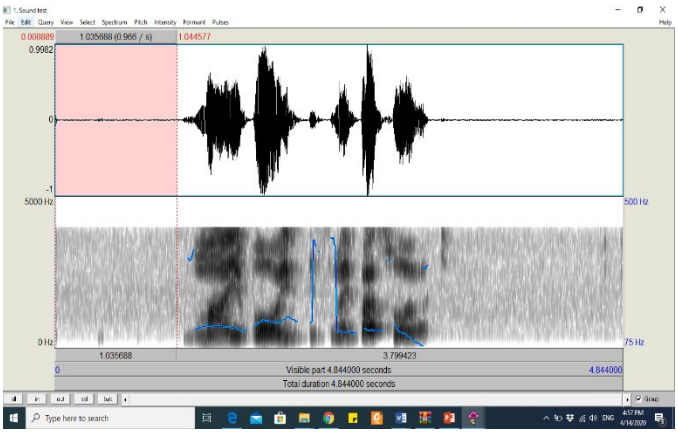

Figure 2. 5. Voice waves before noise reduction

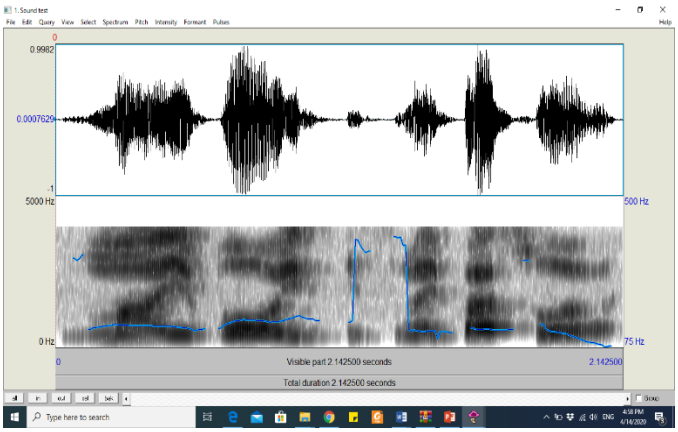

Figure 2. 6. Voice waves after noise reduction

2.7.4 Feature Extraction

In this step, using SPro, the feature extraction process was conducted. The voice data, pre-processed in the preceding step, was subjected to feature extraction. Feature extraction was conducted through 12 MFCCs, as shown in Figure 2.7, and was subsequently stored in a parameter file. The 12 MFCC is about extracting digital features that can be understood by the Alize framework. Alize framework cannot process voice signals instead, it rather takes as input the 12 MFCC features to conduct user voice processing.

The frontend of the system stops at this step and, according to our framework, is accomplished by the user's smartphone. At this point of the testing we used a virtualized Android OS to represent the mobile part of the implementation. See Section 2.7.5 for the next step in the framework which covers training and Section 2.7.6 which covers validation.

```
frame [1]:                              frame [2]:
    numberOfCoefficients = 12               numberOfCoefficients = 12
    c0 = 999.8048601455131                  c0 = 981.7050655028498
    c []:                                   c []:
      c [1] = -44.22161952301026              c [1] = -34.44466435155556
      c [2] = -65.33400756773091              c [2] = -60.544813681496144
      c [3] = -68.47878265501046              c [3] = -56.85040740532389
      c [4] = -77.75213262672578              c [4] = -106.5498611426033
      c [5] = -39.118073561448305             c [5] = -13.863919139306429
      c [6] = -27.97100271788985              c [6] = -52.19559700536588
      c [7] = -16.280815060716662             c [7] = -28.201681724169667
      c [8] = -28.29389137258743              c [8] = -40.52533703197048
      c [9] = -48.891519014221736             c [9] = -78.32348751266473
      c [10] = -21.636367646690204            c [10] = -38.55424495842941
      c [11] = -41.99618380453937             c [11] = -51.86373727536856
      c [12] = -33.930970331179246            c [12] = -32.59291861384919
```

Figure 2. 7. 12 MFCCs features

2.7.5 Voice Model Training / Server Side

In the Mistral toolkit, the next step is training the model. This step controls the list of users of the smart IoT device. Using the Train World process, we used the GMM to conduct this part of the training with the MIT dataset for the UBM training because the procedure requires an already-trained UBM. The remaining 12 were kept for testing. For this part of the implementation we used a virtualized Linux server to host the Android Things operating system, which acts as the manager of the IoT devices.

2.7.6 Verification

After both parts of the model were trained, the data was compared to the world UBM model to create verification. The tests were run twice; the first test included using enrolled speakers. The second test did not include enrolled speakers. After the testing phase started, scores were calculated for each test speech-segment verification based on the Mistral toolkit. By using a decision threshold-speaker model, verification could be either be accepted or rejected. In commercial verification systems, users are required to test and decide on the threshold. However, in our implementation we only speculated on the possible use of the threshold.

2.7.7 Result and Discussion

We had no problem with the enrollment of the speakers in the dataset. In the first test only the intended target speakers were used to train the UBM before they were enrolled. Figure 2.8 shows the detection error tradeoff curve of the MIT voice dataset.

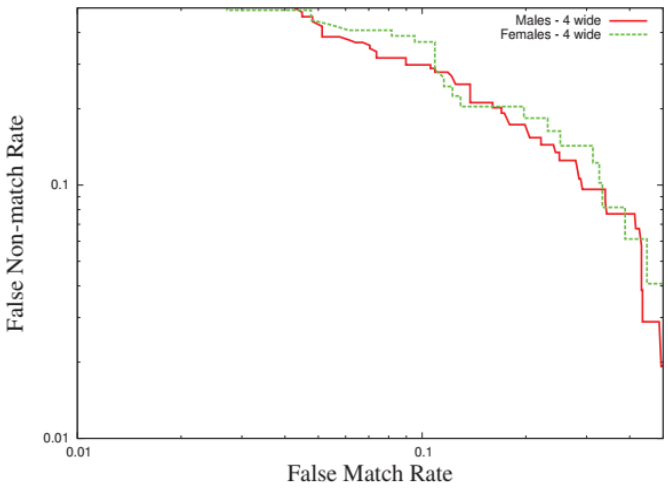

Figure 2. 8. DET plots for a male only trail and female only trail

In the future we will collect enrollment segments from many users anonymously and use different devices in regular daily life settings for a significantly improved result; this will build a more reliable UBM to be distributed for application. In addition, a secure remote authentication mechanism will be

investigated. Voiceprint security, once stored in the voice database, will be included in the future research. The work will focus on the resource-constrained nature of the IoT technology and propose different tools for that aspect. The robustness and resilience of the tools will be thoroughly studied scientifically.

2.8 Chapter Summary

To prevent unauthorized users from accessing the IoT ecosystem, behavioral biometrics authentication systems are considered the most effective. Through voice recognition, it is believed that IoT user authentications will be more secure, accurate, and robust. Our research demonstrates that voice biometric-based authentication models can be used in IoT ecosystems. We proposed and implemented a model that is intended to be used by IoT technology owners to remotely and securely authenticate themselves on smart technologies. Hence, in this chapter we proposed a text-dependent voice recognition system. The system consists of two phases: the enrollment phase, where the user is supposed to enroll the voice, and the verification of authentication phase, where the identity claimer is expected to utter the voice and be subsequently compared with the enrolled one. We tested the model for its usability and have shown a promising result. It is worth mentioning again that we did not include any security measures in our implementation, leaving this for future work. For the sake of repeatability, the coding of the Alize method used in this chapter is included in Appendix A.

Chapter 3: Voice-Feature Learning Using a CNN Designed to Avoid Replay Attacks

Speaker verification is a promising Internet of Things (IoT) user authentication model. However, verifying the identity of an individual using speech utterance is challenging, especially in the presence of replay spoofing attacks, where attackers can record the speech of the user and use it for unauthorized access. In speaker verification, the voice signature is often polluted by ambient noise. This alone confounds the design of appropriate feature extractors that can distinguish the authentic voice signal from that of a replay voice signal. The literature to date reflects very poor performance; notably with state-of-the-art verification errors no less than 11.3% False Positives (FPs) from all samples. To address this difficult gap that has, up to now, prevented speaker verification from being a satisfactory IoT authentication alternative, we are seeking to develop an 80-90% improved approach yielding only 1-2% FPs. To accomplish this goal, we have developed a new approach that uses MFCC and Constant Q Cepstral Coefficients (CQCC) as input features, a Convolutional Neural Network Based (CNN-based) frontend feature extractor, and SVM backend classifier. Using better training techniques such as batch normalizations, we improved the precision significantly. In addition, a formulation that includes knowledge of distinguishing genuine voice from a replay voice was implemented. The experiment was conducted on ASVspoof 2017 datasets [68, 95] (see Section 3.6). An improvement in the state-of-the-art performance was obtained, achieving 7.1 Equal Error Rate (EER) which compares to 11.3 EER in typical state-of-the-art systems.

3.1 Introduction

Speaker verification systems (SVSs) are part of user authentication systems meant to verify the identity of individuals using voice biometrics. Such systems take the voice of an unknown speaker with a claimed identity as input to determine whether the claimed identity matches the actual voice of the claimer [39]. SVSs rely on the recognition of specific, well-learned parameters to authenticate an individual. SVSs analyze recorded voice to determine whether the claimer is genuine or an impostor. In contrast to traditional object-based (non-biometric) systems that use key, smartcard, or password, speaker verification systems remain the most resilient and robust trend of authentication [37]. However, in recent times, SVSs are reported to be vulnerable to manipulations through specialized spoofing techniques such as impersonation, replay, voice conversion, and speech synthesis. Compared to other spoofing attacks, replay attacks are the most straightforward to implement and are highly effective in deceiving speaker verification systems [40].

Conventional speaker verification systems accomplished tasks using GMMs on MFCC feature vectors [41]. The systems also used joint vector and intermediate vectors (i-vectors) to perform speaker-specific representations of a speech utterance [8]. Despite being a state-of-the-art approach, the standards showed shortcomings [17]. For instance, feature vectors suffer from the issue of precision while using MFCC [42]. Likewise, GMM-UBM and i-vector systems have disadvantages because of their unsupervised nature, since they are not trained objectively for speech replay detection. We opted for enriched input data to cope with the challenges. Hence, the hybrid features of MFCC and CQCC are employed to train the feature extractor. With the advent of deep learning in different domains, data-driven approaches using deep learning approaches have also been used by the speaker verification systems [43]. CNN networks have shown a promising result and substantial improvement in speech recognition, computer vision, and related areas due to their ability to deal with real-world, noisy datasets without the need for handcrafted features [44]. For instance, the researchers in [45] employed CNN in SVSs. Comparing their work with traditional baseline systems, they reported an increased precision of 84%.

In this work, we consider that the voice biometrics are acquired from pre-recorded speech, by replaying the device containing the voice, and extracting features that represent the speaker. The dynamics of the utterances of the voice are lost. This may also happen in cases where the speaker is remotely accessing the speaker verification system for enrollment or verification. That is, the enrollment process is accomplished via a phone call or a recorded voice from the speaker. Therefore, discriminatory feature extractors are required for such pre-recorded user voice utterances. In addition, identifying characteristics of the voice that can be used for implementing the feature extractor is difficult. This is a first of its kind, where most studies in this domain have tried to find a representation for the real voice from the speaker; in other words, designing feature extractors for SVSs [19]. Up to now, no feature extractor has emerged in this regard to suit SVSs. The difficulty of finding representations for speech reflects on the classification performance of SVSs, in particular, distinguishing genuine voice and a replay spoof attack targeting a particular person. Considering previous work conducted on large public datasets, the researchers in [46] reported better results, achieving EER around 11.3 %.

To address the issue of obtaining a proper representation for speech with an improved classification performance, a framework for learning the representations directly from a genuine speech using CNN was used. A novel formulation was used that takes into consideration and incorporates techniques to replay spoof attacks using a multi-tasking learning strategy. The model can learn voice parameters in a recorded speech to discriminate between genuine voice and replayed voice. The model has been evaluated to determine whether this feature representation that was extracted using the CNN network

can detect replay spoof attacks. SVSs usually consist of a frontend, where the voice features are extracted, a backend where feature classifiers are used, and a decision where a probabilistic linear discriminant analysis is used. Traditional SVSs use GMM-UMB and i-vector for frontend modeling. The main disadvantage of these models is that they have fallen short of detecting replay attacks. The study focuses is on the frontend, whereby the combination of two feature extractions are employed to train a new CNN architecture. In the state-of-art systems, researchers employed CNN as backend feature classifier; only a few have used it for feature extraction. In our work, we are targeting an end-to-end SVS that employs CNN both in the frontend, as well as a backend classifier. An end-to-end system treats the entire system wholly, as an adaptable black box. The process of feature extraction and classifier training are achieved together with an objective function that is consistent with the evaluation metric. However, in this chapter, we present our frontend implementation which achieves improved precision in detecting replay spoof attacks. In this document, the terms 'replay attack' and 'replay spoof attack' are used interchangeably.

To train our CNN network we first studied existing feature extractions employed in SVSs for replay attack detection. Using single-feature extraction did not work well with replay attack detection. In that light, we opted to combine two low-level feature extractions that included MFCC and CQCC. Therefore, to provide superior accuracy, MFCC was used for processing the high-frequency area of the voice utterance, while CQCC was used for processing silence or areas with low-frequency. The selection between these two feature extractions was made since MFCC is well-known for appropriately capturing voice features' high frequency and that the CQCC is a multi-resolution cepstral feature. The two features were combined as an input to the CNN network architecture implemented in the study. The CNN architecture was implemented by TensorFlow/Keras, as reported in Section 3.5. Once the CNN network was trained, it was employed as the frontend feature extractor. Subsequently, an SVM classifier was trained with the same features to test the performance of the new CNN architecture and the combination of the two feature extractions. The improvement achieved by this work is reported in the following sections of this chapter.

3.2 Speaker Verification System

Speaker verification system (SVS) is the process of recognizing the identity of a speaker based on the verbal utterance of the speaker. The principle behind speaker verification is that every speaker's voice is unique, like fingerprints, and thus can be used to authenticate the speaker. SVSs are used for speaker authentication, surveillance, and forensics. SVSs consist of three phases, as depicted in Figure 3.1. These include the frontend, backend, and decision phases [47]. In the frontend phase, the SVS is trained, using the available input data to learn speaker-specific information from voice signals. In the backend

phase, the speaker's voice is fed into the classifier system to produce the models. Finally, in the decision phase, the speaker model is compared with existing models for similarity. Basically, speaker verification systems use GMM-UBM systems to accomplish these tasks. In the process of implementing the speaker verification system, the Universal Background Model (UBM) was created using MFCC feature vectors. A GMM-UBM is trained on the entire pool of speakers' voice data. Consequently, the trained UBM model is used to calculate the speaker-specific model by adjusting it into individual speaker's data. The UBM is represented as $\lambda_{ubm} = \{w_i, u_i, \sum_i\}C_i$, where $w_i, u_i, and \sum_i$ are weights, mean, and covariance of i^{th}component in C of Gaussian components, respectively. In the enrolment phase the S number of speakers are adapted according to the UBM created from the training data to create speaker models $\{\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_s\}$. The GMM is specified as,

$$P(x/\lambda) = \sum_{c=1}^{C} w_c p_c(x)$$

$P_c(x)$ is represented as,

$$p_c(x) = \frac{1}{(2\pi)^{d/2}|\sum c|} \exp\left(-\frac{(x-\mu_c)^T \sum_c^{-1} (x-\mu_c)}{2}\right)$$

where d is the dimension of the input vector x. The GMM parameters are calculated by using the expectation maximization algorithm. In the evaluation phase, the likely ratio between the target and background models is calculated. The i-vector uses a low dimensional feature vector containing speaker-specific and voice variability information in a voice segment, which is the GMM super-vector by a single variability space. It is written as,

$$M = \mu + T\omega,$$

where u is the mean super-vector, T is the low-rank total variability matrix, and ω is a low-rank vector, referred to as the i-vector. The Probabilistic Linear Discriminant Analysis (PLDA) model is then employed to generate verification scores by comparing i-vectors from the different voice utterances. Some researchers have employed conventional systems, such as GMM-UBM and i-vectors with DNN, for better performance and precision. These systems are usually employed in text-dependent SVSs [48, 49]. For instance, Deep Neural Network (DNN) is used for extracting frames from a speaker's voice and calculating the utterance-level information. Subsequently, the output of the DNN is converted to i-vectors, and, at the backend, PLDA is used for a verification score [50, 51]. In such work, the DNN features are either used alone or combined with the conventional features (i.e., MFCC). Features

extracted from DNN are employed in the posterior. These researchers use DNN to extract speaker-specific data and apply the similarity metrics check for the verification score [45].

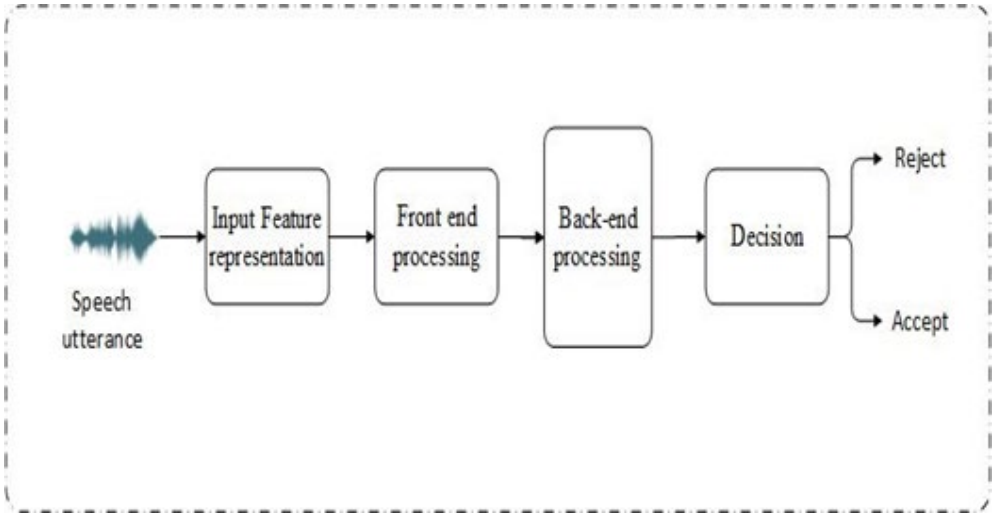

Figure 3. 1. Speaker verification systems

In contrast, some end-to-end usage of the DNN systems can be found in the literature [44]. In these systems, the process of feature extraction and classifier training are accomplished in a way that is consistent with the evaluation metric. Network in Network (NIN) layers are used to construct end-to-end speaker verification systems and have shown considerably good performance [49, 52]. In most cases, such systems are employed in text-independent SVS. However, the systems using DNNs have resulted in improved performance and accuracy, but required many more training samples. That is, the deeper the model the better the accuracy at a cost in terms of training time. Another deep learning approach that has been employed for speaker discrimination is the use of CNNs for feature extraction. From the resource-constrained view of IoT devices, the neural network architectures used so far for SVS may become too big for the memory of IoT devices. They cannot handle the large amount of training data needed to produce such precise and accurate results.

3.3 Deep learning approaches in Speaker Verification Systems (SVS)

This section discusses research that uses a DNN as the primary algorithm for the SVS. More recently, researchers have begun investigating CNNs for speaker verification [53]. For instance, in [54], the researchers have built a speaker recognition system that used senone probabilities computed by a CNN and compared it favorably with an i-vector approach. Chen et al. trained a Siamese Neural Network (SNN) to directly compare two voice segments discriminatively [55]. They relied on MFCC features. In [46], the researchers used CNN to capture and discard speaker and non-speaker information. They used CNNs in the training phase for creating the background model. In a quest to obtain a stand-alone

SVS that can be used with smart home devices, the researchers in [56] employed a small CNN that needed limited training data. The study used transfer learning of representation, whereby first a model was trained with a large dataset and subsequently a model was trained in another task to suit the smart home devices that need a smaller dataset. Nagrani et al. used an end-to-end SVS that used noisy and unconstrained data input [44]. The system involved performing active speaker verification.

Studies that used a CNN for replay attack detection include the those conducted by Huang et al. [57] and Huang et al. [58]. The researchers used a hybrid of MFCC and CQCC features for the extraction of speech particulars to differentiate genuine and replay voices. The study employed CNN as a classifier in the backend of the SVS and used GMM for the frontend feature extractor. In comparison, our approach employs CNN as a frontend feature extractor. For the detection of replay attacks, the researchers in [59] built a frontend feature extractor using CNN. However, in contrast to our work, they employed only CQCC features for the system. In [60], they used a DNN for a frontend feature extractor, employing the High Frequency Cepstral Coefficients HFCC and CQCC features extraction methods and used a Support Vector Machine (SVM) for the backend classifier.

3.4 Feature Learning

In the CNN work described here, we present the formulation of the features selected for speaker verification and evaluate the performance of the features for training classifiers. We first noted that conventional feature learning approaches (machine-learning approaches) directly applied for speaker verification classification are not practical for IoT technologies. That is true because the training data should be very small when used in the IoT context, while most feature-learning algorithms need a large number of parameters for improved performance. In addition, genuine and recorded voice share some properties, and we made use of this similarity for feature learning. Therefore, the system proposed in this study, has a frontend that uses *speaker independent features* followed by *a speaker dependent* classifier. This is necessary to leverage data from users to learn features that capture the intrinsic properties of the user voice. Subsequently, the classifier was trained for the user using these same features.

Compared to other deep learning methods, CNNs are a particularly suitable architecture for speaker verification. The reason is that CNN architecture scale better than fully connected models for larger input sizes and have a smaller number of training parameters; a desirable property for the resource-limited nature of IoT technologies. Thus, to detect a replay attack, a thorough study of the difference between the genuine and recorded voice is paramount. As such, we cannot reduce the voice data too much without the risk of losing the details that enable discriminating between them (i.e., authentic/genuine versus spoofed). A CNN architecture was trained and subsequently the output of a

trained network was used to train the classifier. The idea is that genuine voice and replay voice will be easier to separate by using CNNs for capturing the intrinsic properties of the speech. CNNs are particularly suitable for speaker verification. By having a smaller number of trainable parameters, the architecture used in this work has scales that are better than fully connected methods that require large input sizes, having a smaller number of trainable parameters.

As raised previously, detecting the replay attack requires the system to identify whether the voice segment is played live or recorded-and-replayed. Naturally, as is typical in replay attacks, the tone and context from the voice of the speaker does not change. Hence, learning the very discriminating features to detect the replay from the genuine voice is way too complex. However, since the replay voice uses recording and replay devices, the replayed voice carries information about the two devices on top of the genuine voice. This can be seen in Figure 3.2 as the difference between the genuine and the replay voice is obvious. *Replay attack detection not only needs to focus on the content of the speech, but whether sound characteristics of the recording and replay devices exist.*

It became difficult for one type of feature, such as CQCC, standard MFCC, or Long-Term Average Spectrum (LTAS), to fit in the representation of replay spoof detection [61]. To that end, the hybrid features of CQCC and MFCC were used as an input to the CNN extractor in our approach. Previous studies conducted on the replay attack detection have shown promising results by employing the same hybrid feature extraction method to train the classifiers. The hybrid features work better in determining the differences between the genuine and replay voice [62]. Some researchers have used deep learning for feature extraction but with different combinations of features. For instance, the hybrid features of HFCC with CQCC was used in [60], and researchers in [59] used CQCC for the replay attack detection.

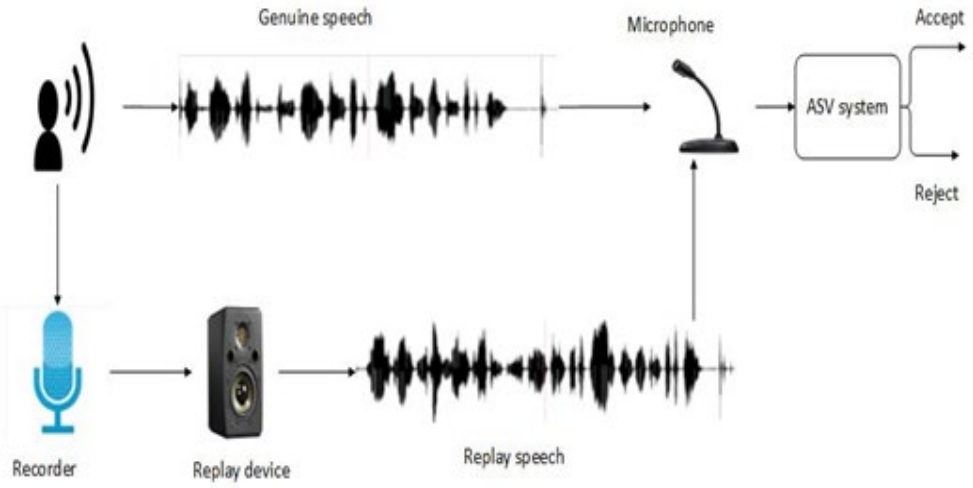

Figure 3. 2. Replay attack process on speaker verification systems

In the extraction of the MFCC features, as one of the best-known features of speech processing, the power-spectrum of the framed voice signal is transformed by a filter bank of dimensionality reduction. In speech processing, this research uses MFCC to process high-frequency regions that are most affected by the spoofing artifacts, while CQCC is used to extract low-frequency regions that may carry minor, yet additional information to distinguish between genuine and replayed voices. These two features are used to exploit possible complementary characteristics in feature space. As shown in Figure 3.3, the resulting hybrid features are used for training the CNN feature extractor. In contrast to the hybrid features employed in [58], the research used zero-mean and unit variance normalization of 30-dimentional MFCC and CQCC features, along with first and second derivatives. The mean and variances were computed on the full training data. The next section provides the CNN architecture used in the feature extraction phase.

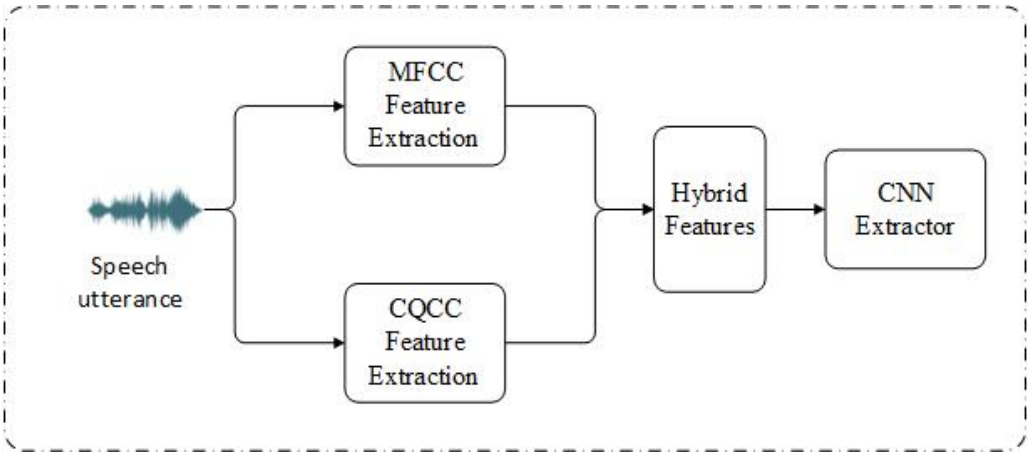

Figure 3. 3. Hybrid feature extraction process

3.5 Front-end CNN Feature Extractor Architecture

In the pre-processing work conducted, this feature extraction process included fixing the input size of the data to suit the Keras environment because the data had varied size. The pre-processing measures taken in feature extraction are discussed in the experiment presented in the next Section 3.6. In the development phase, the CNN was trained with a development set of voice using the formulation defined below. Subsequently, the input voices' data were projected onto the representation space learned by the CNN for the exploitation set. A binary classifier was trained for each voice utterance. CNNs are a suitable architecture for SVSs having a smaller number of trainable data. CNNs can scale better than DNN models that need larger input samples. This is a desirable property for the IoT devices that suffer from resource-limited property. CNNs are also convenient since there is no need to reduce the input data too much, which risks losing the details that enable discrimination between replay and genuine voices. The method is illustrated in Figure 3.4. The development subset D was used to train the CNN.

The result is a function of ϕ. As discussed in Section 3.4, X input voice from the training subset is projected to feature a variation of $\phi(X) \in R^m$, where m is the dimensionality of the projected feature variation. Features learned from the development set D are useful in the separation of genuine and replay voices. The trained CNN was next used as a feature extractor to obtain or extract the feature vector $\phi(X_E)$ for each voice utterance from the training set (E). The new representation is then used to train the classifier f. For a new voice utterance X_F from the evaluation set F, the CNN was first used to extract features to obtain the feature vector $\phi(X_F)$. Subsequently, this extracted feature vector was fed into the binary classifier to obtain the final decision $f(\phi\ (X_F))$.

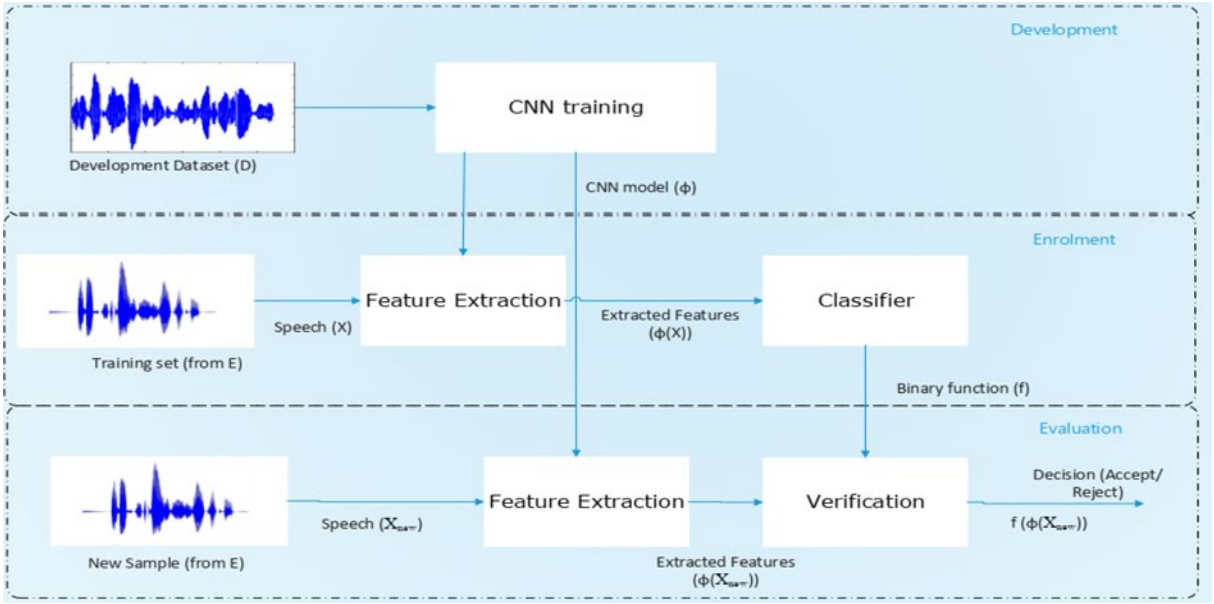

Figure 3. 4 CNN architecture for my feature learning approach.

An architecture similar to VGG-M [62], mostly used for image classification and speech-related application, was employed. Table 3.1 shows the details of the CNN layers. Two fully connected layers after convolution layers were used. Softmax activation with N neurons were used. N is the number of speakers in the development set D. Each convolution layer was followed by a pooling layer that generated a lower resolution version of the convolution layer's activations by taking the maximum filter activation. This ensured some degree of shift and distortion invariance. Stride is the distance between applications of the convolution or pooling operation. Rectified Linear Units (ReLUs) were used as the activation function for all convolution and fully connected layers, except the last one, or the output layer that uses Softmax. The model weight was initialized according to the work in [63]. Once the CNN had been trained in set D, it was used for feature extraction. The model is trained with Nesterov Momentum for 60 epochs, using a momentum rate of 0.9, and mini batches of size 100. The networks were implemented with the libraries, TensorFlow [64] and Keras [65]. It took four hours to train on the

GPU Tesla C2050. With the CNNs, we were able to circumvent the necessity of strongly pre-processing the raw audio data and thereby losing possible valuable information. The network was allowed to choose the necessary features, especially for the verification task, from a wide range of available possibilities. The network was trained from end-to-end, starting from spectrograms as in [64]. To prevent over-fitting, a dropout layer between the layers was used. That is, to address the over-fitting problem, a dropout layer is inserted into the all-connected and output layers, to randomly drop units and their connections from the CNN network.

Table 3. 1 Summary of the CNN architecture and layers

Layer	**Kernel**	**Stride**	**Size**
Convolution 1	7 × 7	1 × 1	363×190×17
Convolution 2	1 × 1	1 × 1	440×210×21
Convolution 3	3 × 3	1 × 1	440×210×36
Convolution 4	1 × 1	1 × 1	216×100×48
Local response 1	5 × 5	-	-
Local response 2	5 × 5	-	-
Pooling 1	-	2 × 2	363×190×10
Pooling 2	-	2 × 2	440×210×18
Pooling 3	-	2 × 2	216×100×32
Fully Connected + Dropout	-	-	4096
Fully Connected + Softmax	-	-	N

Subsequently, employing the same transfer learning as in [56], the representation was obtained by performing forward propagation. Based on the formula, after forward propagation, for the input X, the feature extractor function $\phi(X)$ is the representation network at the last layer before Softmax. This makes the features, learned for the set D during the training of the CNN network, stand relevant for classifier training. For training, the classifier, the training, and validation sets were used. For each speaker utterance in E (the training set), the trained CNN extractors were used to extract features, and the extracted features were submitted to train the classifier f. For the classification, SVM were tested (both linear SVMs and SVM with a RBF kernel) [66, 67].

To train the classifiers, feedforward propagation was performed until the last layer before Softmax and thus used the activations at the layer as the feature vector for the speech. This was achieved by using a transfer learning between sets of users. For each user, a training set consisting of genuine and replay voices were built. The SVM was trained both in linear formulation and with the Radial Basis Function (RBF) kernel. Likewise, for the evaluation, a disjoint set of genuine and replay voices that had not been used for the training were used.

3.6 Experimental Setup and Results Validation

In the research, we used the ASVspoof 2017 dataset [68] used by previous researchers for comparison. The dataset was created by the RedDots project [40], and consists of audio files of different speakers and different environments. The dataset contains three sub-datasets, including training, development, and evaluation, which contains 3014, 1710, 13306 audio files, respectively. The audio sets are tagged as genuine or spoofed voices. The frontend feature extractor was trained and verified with these tags. The summary of the dataset is detailed in Table 3.2. The dataset was prepared for replay spoof attack. The replayed data was collected from a pool of volunteers with access to a diverse array of replay and playback devices, in different session conditions. Audio signals in the dataset have a resolution of 16 bits, a sampling rate of 16 kHz, and utterance durations from 3 to 5 seconds. The size was large enough to allow details from signals in the voice, while also small enough to enable training. While there was no overlap in speakers between the data partitions, there was some overlap in replay configurations between the partitions.

Table 3. 2. Summary of dataset used in the experiment

Subsets	#Speakers	#Genuine utterances	#Spoofed utterances
Training	10	1508	1508
Development	8	760	950
Evaluation	24	1298	12008
Total	42	3566	14466

For pre-processing, the speech pad was cut to 3 seconds. Subsequently, in the feature extraction process, a hamming window structure was used for both MFCC and CQCC framing setups. The size of the window was set at 50ms and the step size of 25ms with a 25ms overlap. The minimum frequency was set to 3 kHz and maximum frequency was set to 8 kHz. After the features MFCC and CQCC were extracted, the two were combined to train the network. In the experiment, it was evident that the employment of CNN as the feature extractor in speaker verification is more precise and outperforms conventional systems, such as GMM-UBM and i-vector, in detecting replay attacks. As reported in a number of research papers, employment of deep learning is advantageous. In [46], the study compares CNN and GMM-UBM using Equal Error Rate (EER): GMM-UBM with an EER of 17.1, i-vector with an EER 12.8, and CNN with 11.3 EER. In [55], which employed DNN for speech, feature extraction with better performance compared to GMM-UBM and i-vector was obtained. However, our work shows better precision and performance compared to those researches. That is, the experiment has shown better replay spoof detection in comparison to the systems reported in the literature. Hence, with

the two different features used for classification, we obtained EER of 7.91% with the leaner SVM, and 7.1% using the EBF kernel.

3.7 Chapter Summary

A three-layer SVS that uses CNN, based on text-independent feature learning, has been presented. The method does not rely on direct utterance by individuals; rather, a recorded sample speech was used for differentiating genuine speech and forged speech. The experiment was conducted on the ASVspoof Dataset (see Section 5) to demonstrate that this method is promising; it achieved better performance when compared to state-of-the-art techniques presented in the replay attack challenge 2017. In the experiment, we were able to train CNN for feature extraction and SVM for feature classification. The results show a promising improvement. The CNN network and SVM coding employed in this chapter can be found in Appendix B and Appendix C, respectively.

Chapter 4: Replay Spoof Attack Detection Using DNNs for Classification

In this chapter, we explore the use of the deep learning-based backend in speaker verification systems (SVSs) for replay spoof detection. Automatic speaker verification (ASV) systems can be easily spoofed by previously recorded genuine voice. To counter the issues of spoofing, detecting replay spoof attacks play an important role. Hence, we consider the detection of replay spoof attacks the most easily accomplished spoofing attack. In this light, we propose a Deep Neural Network-based (DNN) classifier using a hybrid feature from MFCC and CQCC. Several experiments were conducted on the latest version of the ASVspoof 2017 dataset. The results are compared with a baseline system that uses the Gaussian mixture model (GMM) classifier with different features that include MFCC, CQCC, and the hybrid features of the two. The experiment results reveal that the DNN classifier outperforms the conventional GMM classifier. It was found that the hybrid-based features are superior to single features, such as CQCC and MFCC, in terms of EER. In addition, as other researchers have discovered, high-frequency regions of voice utterance convey greater discriminative information for replay attack detection.

4.1 Introduction

The use of biometrics for human authentication has become an important part of securing applications [69]. Although, in the last decade, the world has experienced rapid adoption and increased effectiveness of biometrics-based authentication systems, the success of such biometric systems is not only measured by its effectiveness, but also by the availability and affordability of the technology required to measure the human factors it uses [70, 71]. Voice is one of the most commonly used biometrics in the market nowadays, because it has the advantage that it can be easily captured with readily available personal devices [18]. Hence, ASVs have gained the attention of both researchers and industry [72]. Nevertheless, ASVs are vulnerable to spoofing attacks where an impersonator claims the identity of someone that he is not [73]. Known spooking attacks associated with ASVs include mimicry attacks, conversion attacks, and replay attacks. A mimicry attack is when a professional imitator mimics the utterance of the voice of a genuine user to gain unauthorized access [74]. Conversion attacks happen when voice transformation is performed on an imposter's utterance so that it sounds more like that of a genuine user [75]. Finally, a replay attack is a spoof technique where pre-recorded speech is provided to speech verification systems, with the help of record-and-play devices, to impersonate a genuine user [73]. Among the three spoof attacks, replay attacks pose the most serious threat to ASVs [40]. The technique does not require expertise and is difficult to detect (i.e., enjoys more than 50% success rate). A high-quality smartphone is sufficient for someone to employ the technique effectively. As a consequence, there is a pressing demand for robust replay attack detection methods [40, 76].

Some traditional replay prevention attack techniques exist, such as the use of text-prompted ASV systems [77, 78], and the use of additional biometric modalities [79, 80], for use as a means of replay attack prevention. In these systems, there is a tradeoff between system security and system complexity [81]. Therefore, replay attack detection approaches have attracted researchers [82, 83]. The replay attack detection methods are of two approaches. One approach researchers have opted for adds auxiliary information to the utterance; for instance, researchers in [84] proposed the addition of a time stamp in the utterance of the genuine user, thereby validating the time stamp in the replay attack. The other approach focuses on detecting the distortion associated with the recording and replay devices that are used to accomplish the replay attack [40]. This latter approach is where we choose to focus our attention for the detection of replay attack.

Several researchers have worked on the detection of distortion as a countermeasure to replay attacks. For instance, in [85], to detect the replay attack the research found that the spectral ratio, low-frequency ratio, and modulation index carry relevant information to differentiate between the replay speech from the genuine speech. A matching algorithm that detects the cut and paste of the voice is employed and the detection is achieved by comparing the pitch and MFCC contours using Dynamic Time Warping (DTW). Several replay detection solutions on text-independent ASV systems are proposed in [86]. Conventional ASV systems were employed in this research. However, in later studies it became apparent that such systems have issues of precision and performance when detecting replay attacks. For example, in [61], the research proved that, with the use of replay attack detection GMM-UBM-based ASV systems, the equal error rate (EER) increased from 2.61% to 15.03% for 459 genuine male and 1.82% to 34.09% for 220 genuine female speeches. Hence, it can be concluded that GMM-UBM-based systems are highly vulnerable to replay attacks [87]. In this light, domain research started employing deep learning techniques to overcome the problem [88]. Neural networks for ASV systems provide more natural solutions than conventional algorithms.

This inspired the successful application of deep learning frameworks in the ASV systems domain for replay attack detection. For example, some of the researchers have employed conventional systems, such as GMM-UBM and i-vectors with DNN, for better performance and precision. These systems are usually employed in text-dependent speaker verification systems [48, 49]. For instance, DNN is used for extracting frames from a speaker's speech and calculating its utterance-level information. Subsequently, the output of the DNN is converted to i-vectors, and at the backend, PLDA is used for a verification score [50, 51]. In such work, the DNN features are either used alone or combined with the conventional features (i.e., MFCC). Features extracted from DNN are employed in the posterior. These researchers use DNN to extract speaker-specific data and apply the similarity metrics check for the

verification score [45]. However, there is always room for improvement, and such domain researchers are still working on obtaining systems and methods with better precision and performance that use deep learning techniques. There are several studies using the deep learning model for replay attack, and it remains a hot area of research. Different studies using deep learning models for replay attack detection are presented in the related work section below.

In this chapter, we propose a deep neural network-based classifier that uses hybrid features. Our contribution is twofold. The first is at the feature selection level, where we propose a hybrid feature that uses CQCC and MFCC to capture discriminative characteristics of both high-frequency and low-frequency regions of speech utterance, respectively. The second is that we propose a model-level work that contributes a DNN-based classifier that is trained to discriminate between different conditions due to changes in playback, recording, and environmental characteristics and context.

4.2 Modeling in Speaker Verification

The conventional SVSs used GMM-UBM and i-vector as state-of-the-art approaches of modeling for a long time. The modeling approaches rely on dimensional input features for extraction using MFCC. Nevertheless, MFCC has shown performance degradation [89]. As a result, domain researchers have started investigating deep learning approaches for better feature extraction [43, 45]. Deep learning approaches have shown promising results with other verification systems using biometric trails, such as face and signature. For instance, DNN gained momentum for use in re-identification problems in face recognition [90] and the identification of a forged signature in signature verification systems [91]. Hence, using DNN, CNN, or Recurrent Neural Network (RNN) for feature extraction of biometric models has become a trend [59]. Particularly, DNN and CNN approaches have become more suitable for the extraction of speaker-specific features to detect replay spoofing attacks [55].

Several replay attack detection approaches have been proposed following the ASVspoof attack challenge held in 2017 [68]. Lantain et al. [92] observed that the use of Inverted Mel-frequency Cepstral Coefficients (IMFCC) for replay attack detection outperformed that of standard MFCC features. In [60], the researchers proposed a DNN-frontend architecture, which uses HFCC in tandem with CQCC features, for replay attack detection. They employed the ASVspoof dataset and used a SVM classifier. Some of the research has employed the conventional systems, such as GMM-UBM and i-vectors with DNN, for better performance and precision. These systems are usually employed in text-dependent SVSs [48, 49]. For instance, DNN is used for extracting frames from a speaker's speech and calculating its utterance-level information. Subsequently, the output of the DNN is converted to i-vectors, and at the backend, PLDA is used for a verification score [50, 51]. In such work, the DNN features are either used alone or combined with the conventional features (i.e., MFCC). Features extracted from DNN are

employed in the posterior. These studies use DNN to extract speaker-specific data and apply the similarity metrics check for a verification score [45]. In [93], the researchers used a DNN-based classifier using features extracted from the LTAS. In the end, the proposed DNN classifier was compared with the GMM classifier that uses MFCC and CQCC features. The result showed that the DNN outperforms the GMM classifier.

The researchers in [59] proposed an end-to-end deep learning approach for the detection of replay spoof attacks. The approach consists of a Lite Convolutional Neural Network (LCNN) (at the frontend) and RNN at the backend. Observations reported in this chapter include that frequency range analysis for replay attack detection performs better with 4-8 kHz and 6-8 kHz sub-bands, which carry the most discriminative information segments. The researchers in [57, 58] have also employed CNN for replay attack detection using a hybrid feature of MFCC and CQCC extractions. They used the CNN as the classifier in the backend, while using GMM for the frontend of an automatic SVS. The researchers built a frontend feature extractor using CNN; they used CQCC features for the system. In [94], the researchers proposed a system with an improved spoofing detection approach. The approach uses Gated Recurrent CNN as a deep feature extractor, which is later used as the backend classifier. They proposed a new input feature, signal-to-noise masks (SNMs). An end-to-end system that employs CNN as feature extractor and RNN as the classifier was proposed for spoof detection in [88].

4.3 Features for Replay Attack Detection

Detecting a replay attack requires systems to identify whether the speech segment is played live or recorded-and-replayed. Naturally, in replay attacks, the tone and context of the speech of the speaker does not change. As depicted in Figure 4.1, the difference between the genuine and replay speeches is obvious. Since the replay speech uses recording and replay devices, the replay speech carries information about the two devices on top of the genuine speech. As such, replay attack detection not only needs to focus on the content of the speech, but also whether the sound characteristics of the recording and replay devices exist. It became difficult for one type of feature, such as CQCC, standard MFCC, or LTAS, to fit in the representation of replay spoof detection [61]. To that end, the hybrid features of CQCC and MFCC were used as an input to the DNN classifier in this research. Previous studies conducted on replay attack detection have shown promising results by employing the hybrid feature extraction method to train the classifiers [93]. The hybrid features work better in determining the differences between the genuine and replay speech. Some researchers have used the learning for classifiers but with different combinations of features. For instance, the hybrid features of High-Frequency Cepstral Coefficients (HFCC) with CQCC is used in [60], and researchers in [59] used CQCC for replay detection.

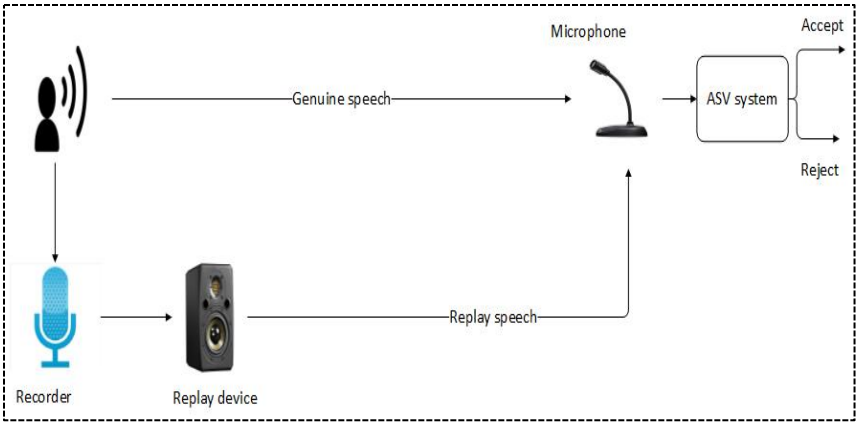

Figure 4. 1. Replay attack process on ASV systems

In the extraction of the MFCC features, as one of the best-known features of speech processing, the power-spectrum of the framed speech signal is transformed by a filter bank for dimensionality reduction. In speech processing, this research uses MFCC to process high-frequency regions that are most affected by the spoofing artifacts, while CQCC is used to extract low-frequency regions that may carry minor, but additional information to distinguish between genuine and replayed speeches. These two features are used to exploit possible complementary characteristics in feature space. As shown in Figure 4.2, the resulting hybrid features are used for training the DNN classifier. In contrast to the hybrid features employed in [58], our research used zero-mean and unit variance normalization 30-dimentional MFCC and CQCC features, along with first and second derivatives. The mean and variances were computed on the full training data.

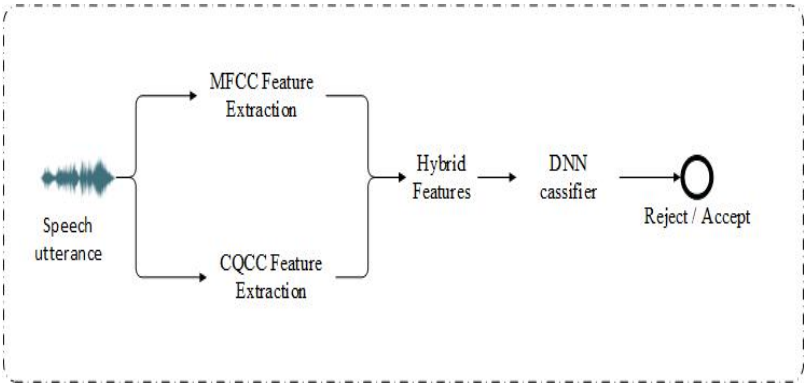

Figure 4. 2. Hybrid feature extraction process

4.4 The DNN-Architecture Classifier

A classification model in the form of DNN is proposed. In most cases, for replay spoof attack detection, DNN was used for feature extraction. The proposed DNN architecture was fully connected to a feed-forward neural network. It consisted of multiple hidden layers. When using deep learning, it is

necessary to have enough samples to improve the performance of the deep network. There are several problems associated with DNN. The first problem is the need for more computational capacity; this occurs from the input of every layer and the parameters of preceding layers that change in every training epoch. The second problem associated with DNN is over-fitting. Nevertheless, these issues can be tackled with batch training (i.e., dividing data into groups), which helps overcome the memory requirement. Batch normalization is also useful to accelerate the learning process and help reduce the effect of hyper-parameters. In the end, the dropout technique solves the over-fitting problem.

Spoofing-discriminant DNNs with five hidden layers were used to distinguish the genuine from the replay spoof speech in this experiment. Each of the layers has 2048 nodes with a sigmoid activation function. The nodes at the output layer were K + 1 and the Softmax activation function was used in that layer. The K + 1 output nodes correspond to one genuine speech and K spoofing attacks. Batch normalized super vectors Fi that are composed of n hybrid features vectors were used for DNN training.

$$Fi = \{x1, x2\ldots\ldots, xT\}$$

Where T is the number of hybrid features. Using TensorFlow/Keras, the DNN was built and trained. Stochastic gradient descent methods were used, and the cross-entropy function was selected as the cost function. The maximum training epoch was chosen as 120, while the mini-batch size was set as at 128. By counting how many frames were similar to the genuine speech, the replay speech was detected. The posterior obtained at the output layer of the DNN was transformed into Log-Likelihood (LLR) score as:

$$LLR_{DNN} = log\ p(genuine|F_i) - log\ p(replay|F_i)$$

4.5 Experimental Evaluation and Results Validation

The research used the ASVspoof 2017 dataset [68, 95] in the experiment. The dataset is produced for spoof attacks, most of which are replay spoof attacks. The resolution of all the audio signals in the dataset is 16 bits; the sampling rate is 16 kHz. The duration of the utterance lasts approximately 3 to 5 seconds. The summary of the dataset is depicted in Table 4.1.

Table 4. 1. Summary of the dataset used in the experiment

Subsets	#Speakers	#Genuine utterances	#Spoofed utterances
Training	10	1508	1508
Development	8	760	950
Evaluation	24	1298	12008
Total	42	3566	14466

A hamming window with the size of 10 ms was used to calculate the short-time zero-crossing rate and short-term energy. Then the threshold was set to find the silent segment and speech segment. The silent segment was of length 512 ms, while the speech segment had a length of 1525 ms. A segment with insufficient length was duplicated. The silent segment was represented with a MFCC feature and, after setting at a frequency of 3-8kKz, it obtained 60*78 features. The speech segment extracts were represented with a CQCC feature and the selected hamming window. The window size was 50ms; and the step size was 25 ms, that is, there was a 25 ms overlap, and 48 coefficients were selected.

After CQCC and MFCC features were extracted and combined, a feature of 60 x 126 was obtained. The feature was then used as an input to the DNN classifier. All three subsets were employed for the model training. The result has shown that the hybrid features with the DNN classifier produced the best performance. We used a single feature as the input to baseline GMM system and DNN system as well. The results are shown in Table 4.2.

Table 4. 2. Evaluation results of the DNN based backend classifier using hybrid features

Systems	**EER%**		
	Dev	**Eval**	**Eval + Dev**
CQCC → GMM	8.18	27.79	15.54
MFCC → GMM	5.54	25.54	14.78
CQCC → DNN	**10.8**	**23.57**	**10.8**
MFCC → DNN	7.36	20.84	13.56
MFCC + CQCC → GMM	8.67	22.67	18.25
MFCC + CQCC → DNN	**2.68**	**7.65**	**5.64**

We used CQCC as input features to the DNN. The system produced excellent results with EERs of 10.8%, 23.57%, and 10.8% on the Dev, Eval, and Dev+Eval sets, respectively (third row blue). Likewise, we used MFCC as input features to the DNN and also showed excellent ERRs of 5.54%, 25.54%, and 13.56% on the Dev, Eval, and Dev+Eval sets, respectively (row four). This shows that our

DNN classifier outperformed the GMM-based classifier. Using the hybrid features (MFCC+CQCC) as input to the DNN system, the result produced ERRs of 2.68% on Dev set, 7.65% on Eval set, and 5.64% on Dev+Eval set (last row). The result also shows the benefit of combining CQCC and MFCC features. The overall scores indicate that the use of the Hybrid features with our DNN system outperforms the GMM systems.

4.6 Chapter Summary

In this study, we explored the application of deep learning frameworks as a classifier for the detection of replay spoofing attacks in automatic SVSs. The study investigated implementation of DNN at the back of ASV systems, which is not common in the literature. The model was shown effective in the experiment and the proposed solution outperformed existing classifications. The ASVspoof 2017 dataset was employed for the experiment where hybrid features of CQCC and MFCC extractions were used to train the model. The hybrid features used as the input to the DNN model showed that good discriminative characteristics were extracted from the speech utterance to differentiate between genuine and replay speeches. The coding of the DNN architecture can be found in Appendix D.

Chapter 5: A Security Hardened End-to-End Lightweight IoT Voice Authentication System

Prevalent use of smart IoT devices and systems combined with home automation demands that such devices play their role dependably and securely from registration through to decommission (i.e., end-to-end). Although password-based authentication systems have the advantage of great convenience, they became exceedingly vulnerable to the point that their length requires a computer to remember. Biometric-based authentication, especially as a second factor, is an obvious and usable tactic to authenticate IoT devices. Herein, we present a secure voice biometric-based authentication system. The basic motive for the proposed voice-based authentication system is to distinguish the legitimate IoT-device user from an imposter (e.g., replay attack). Our approach uses deep learning (i.e., convolutional neural nets [CNNs]) to achieve this objective, namely end-to-end security. We describe a method where the final max/average pooling layer was replaced with a special pyramid-pooling layer in a residual network architecture; this has the advantage of allowed us to relax the fixed-length input constraint and to train the entire network with arbitrary sized voice inputs. As a result, we were able to identify the legitimate users from their imposters. The proof-of-concept has been implemented on a mobile IoT cloud. Control of the system resides in the cloud and the user's smartphone has an end-user app from which the user authenticates themselves to an Android wearable device on the other end. The accuracy of the model has been thoroughly tested and verified repeatably. Also, these experiments demonstrate how easily users can authenticate themselves. The CNN architecture used here was compared to I-Vector baseline systems. The result has showed a significant 15.8% improvement over the baseline system.

5.1 Introduction

In recent years, the Internet of Things technologies have increasingly emerged with the potential for transforming every context of computing along with the expanding interconnection of devices via Internet. IoT technologies are used in a range of applications that include the manufacturing industry [96], smart cities (smart home, healthcare, smart vehicle) [97, 98], and transportation [99]. It was reported that in 2020 nearly 38 billion devices would be connected to the Internet [100]. However, the rising adoption of IoT technologies came with increased security and privacy concerns. IoT technology vendors and manufacturers do not keep in mind the security aspects of their devices. In [102], Gartner predicted that by 2020, 25% of attacks would be on IoT technologies; thus, IoT technology-using organizations or individuals were advised to invest in securing data. In most cases, IoT technologies contain actuators, sensors, or smartphones. Such smart devices possess the capability of sensing, monitoring, and collecting data about the user environment. Such data may contain a user's private or

confidential information. This has increased the attack surface of the IoT devices. Hence, for users to have proper control over their devices, certain applications for smart devices are being developed to access services offered by using IoT networks. For example, Lockitron is a smart device that can lock and unlock doors via remote control, typically through a smartphone. Most IoT technologies can be monitored and controlled via smart device applications [12]. For another example, Voice Control Systems (VCS), such as Siri and Alexa, are used for giving commands to IoT devices to perform certain activities; VCS systems are also used for managing home electrical appliances [101]. Hence, this increases the amount of data stored in such smart devices used for controlling IoT technologies. The huge data accumulated in such smart devices in turn brings with it adverse underlying effects, such as the invasion of privacy and the potential misuse of information about the user [102, 103].

Users remotely access IoT devices through smart device applications to connect to nodes or sensing elements in an unattended manner. Once connected, the user can access desired information from specific nodes. This makes remote user authentication very critical in IoT networks; only legitimate users should be able to access IoT nodes while using any service on his/her smart devices. Insufficient user authentication has been raised as one of the main security issues in the IoT environment. The increasing connectivity features of IoT devices also create another layer of security threat. For instance, smartwatches used to have limited features, such as Bluetooth, but now almost all brands of smartwatches have connectivity features like Wi-Fi. As more connectivity features are introduced to wearable devices, these devices experience more attacks [104].

One of the most commonly used user authentication methods is the use of passwords due to its simplicity. However, in this overly connected world this authentication method creates problems: in addition to having a problem remembering the correct password for various devices and utility services, the user is expected to have at least 19 different passwords to achieve a certain level of strong user authentication [105]. This is the reason why most smart devices have added some kind of biometric-based authentication. Nevertheless, even the biometric-based authentication methods have seen spoofing attacks. For example, fingerprint-based authentication can be easily deceived by high-resolution fingerprint images. Iris-based authentications can be attacked by using high-quality iris images [12]. Other authentication methods currently on market include PIN, smart cards, and token-based authentication. One common bottleneck for all of these authentication methods is that they all need some sort of hardware. One key characteristic of IoT systems is that they are computing, and storage compared to conventional networked systems. Using voice for command has become a popular user authentication in the IoT environment. Even though the voice-based authentication method has the advantage of great convenience, it is extremely vulnerable to spoofing attack due to its broadcast nature

[106]. This kind of attack has opened a door for perpetrators who want to compromise user information and perform undesirable operations, which may pose serious threat to the security of IoT systems.

The spoofing attacks that make voice-based authentication vulnerable include the replay attack. In the replay attack, an adversary tries to use the pre-recorded voice of the legitimate user to compromise the system [73]. One approach that has been proposed to defend against such spoofing attacks is by detecting the recorded voice from the original voice of the user. Hence, a voice-based authentication is measured by its level of detection of the pre-recorded voice as compared to the genuine voice of the user [40, 76]. However, the existing voice-based authentication schemes suffer from issues of precision when detecting replay attacks. As a result, various feature extraction and classification methods are needed to improve their level of precision. In addition, these methods also impose unacceptable risks to privacy since a user's voice data can be utilised to infer the user's behavior in their daily life [81].

In this chapter, we present a secure lightweight voice authentication method based on an end-to-end deep learning feature extraction and classification approach. Our method consists of a speaker-independent CNN feature extraction using frontend, and feature classification using backend. For the feature learning, speaker-independent features are learned using a subset of users, and subsequently used to train speaker-dependent classifiers for another set of users. The adapted CNN architecture was fed with fixed-length voice representation of 3 seconds. The experiment was conducted on the newly released ASVspoof19 dataset that covers various kinds of spoofing attacks, such as voice conversion, text-to-speech, and replay attacks [40]. Conversely, ASVspoof17 focused instead upon replay spoofing attacks and countermeasures. We selected ASVspoof19 because replay spoofing attacks within a physical access (PA) scenario are generated through carefully controlled simulations that support much more revealing analysis than in ASVspoof17 [40]. Our main intention was to propose a secure lightweight voice-based authentication system. Subsequently, its implementation and efficiency is presented. Using the proposed architecture, compared to the state-of-the-art and conventional machine-learning methods, a better performance has been obtained with the end-to-end deep learning methods.

5.2 Deep learning approaches in Biometric authentications

A number of spoofing attacks have been identified in voice authentication systems [107]. There are four main spoofing attacks for speaker verification systems (SVSs), and they include impersonation, replay, speech synthesis, and voice conversion. Impersonation is said to not be a serious threat for the authentication systems. Conversely, the replay attack is the most serious among the other three attacks. Different research has been conducted on mitigating such attacks. To enhance the security of ASV systems against these attacks, automatic speaker verification spoofing and countermeasure (ASVspoof) initiatives have provided a platform for researchers to follow-up, study, and compare spoofing detection

systems [95]. While dealing with speech synthesis and voice conversion require expertise and specialized equipment, the replay attack does not require any expertise or specialized equipment [2]. Many researchers have proposed defense mechanisms for replay attacks [61, 82, 85].

Some researchers have employed deep learning methods as a better way of detecting replay attacks. Lantain et al. [92] observed that the use of IMFCC for replay attack detection outperformed that of standard MFCC features. In [60], the researchers proposed a DNN-frontend architecture, which uses HFCC in tandem with CQCC features, for replay attack detection. They employed the ASVspoof dataset and used a SVM classifier. Some of the research has employed the conventional systems, such as GMM-UBM and i-vectors with DNN for better performance and precision. These systems are usually employed in text-dependent SVSs [48, 49]. For instance, DNN is used for extracting frames from a speaker's speech and calculating its utterance-level information. Subsequently, the output of the DNN is converted to i-vectors, and at the backend, PLDA is used for a verification score [50, 51]. In such works, the DNN features were either used alone or combined with the conventional features (i.e., MFCC). Features extracted from DNN were employed in the posterior. These researchers used DNN to extract speaker-specific data and applied the similarity metrics check for the verification score [45]. In [93], the researchers used a DNN-based classifier using features extracted from the LTAS. In the end, the proposed DNN classifier was compared with the GMM classifier that uses MFCC and CQCC features. The result showed that the DNN outperforms the GMM classifier.

The researchers in [59] proposed an end-to-end deep learning approach for the detection of replay spoof attacks. The approach consisted of LCNN (at the frontend) and RNN at the backend. Observations reported in this paper included that frequency range analysis for replay attack detection performed better with 4-8 kHz and 6-8 kHz sub-bands, which carry the most discriminative pieces of information. The researchers in [57, 58] also employed CNN for replay attack detection using a hybrid feature of MFCC and CQCC extractions. They used the CNN as the classifier in the backend, while using GMM for the frontend of an automatic SVS. The researchers built a frontend feature extractor using CNN; they used CQCC features for the system. In [94], the researchers proposed a system with an improved spoofing detection approach. The approach uses Gated Recurrent CNN as a deep feature extractor, which was later used as the backend classifier. They proposed a new input feature, Signal-To-Noise Masks (SNMs). An end-to-end system that employs CNN as feature extractor and RNN as the classifier was proposed for spoof detection in [88]. In [108], the researchers proposed a voiceprint-based authentication system that used DNN to extract features that can be used alongside MFCC to improve the SVSs. However, the model did not target the resilience of the system in regard to spoofing attacks. In [109], the research investigated a voice pathology-using authentication system that also used

a deep learning approach. The model has been used in a mobile healthcare framework where the mobile device is used for capturing the voice features. The voice signals are pre-processed before they are fed in a CNN network. Using the Caffenet and VGG-16 variant of the CNN network, the result of the experiment showed a voice pathology detection accuracy of about 97.5%.

Biometric-based authentication systems that did not use voice include, for example, the researchers in [112] who proposed a fingerprint-using authentication system that used a deep learning LSTM that enables IoT devices to a extract s set of stochastic features to effectively authenticate the reliability of signals. In [110], an iris-based IoT authentication system that uses a deep learning approach was proposed. The research compared the proposed systems to that of some existing traditional iris-based authentication systems where the proposed systems have shown significant performance improvement. The systems were particularly able to challenge mimicry attacks. Likewise, a selfie face and ocular biometric-based authentication model that uses CNN networks was proposed in [111]. The CNN architectures were used for feature extraction while SVM were used for classification. The two biometric features were collected independently and later fused with different CNN networks, such as VGG and InceptionNet. The model focused on performance and has not been evaluated for its resilience. An IoT-authentication model that used gaze and touch biometrics used for mobile IoT devices was proposed in [112]. In [113], the researchers proposed a deep learning-based authentication model that uses LSTM of unit length 2048 with an accuracy of 99.58%. In [110], the study improved an iris-based authentication model that uses deep learning, training the network with an augmented database. The model achieved promising results of 98.55% for the test set Bath 800 and 99.71% for CASIA. In addition, a CNN-based multi-class authentication model trained over 56 000 iterations, which achieved 99.10% for detecting forgery was proposed in [114].

5.3 Motivation

Authentication systems are used for the validation of the identity of an entity based on a previously registered password. Resilient authentication approaches tend to prevent an imposter, who uses attacks such as a replay attack, an impersonation attack, or a key stolen attack, from accessing a system. In the case when the system fails to identify the imposter from the genuine user, the system can be compromised and private user data, including credentials, may be stolen. In addition, once the imposter gains access to the system, other security threats can be launched to further ruin the system [115]. As such, authentication is the prime security attribute for IoT devices. Since, IoT technologies can exchange data with each other without the consent of the owner, IoT devices accumulate sensitive private user data. Furthermore, illegal access to an IoT device may even be to life threatening; such is the case, if the body-area network device that monitors the health of a patient is compromised, causing

unwanted consequences to the patient. Therefore, authentication must be the first priority when setting up safe and reliable communication among different IoT devices. Authentication is considered as an attractive and favorable approach for securing smart home devices [116].

5.4 Proposed Voice Authentication System (VAS)

The proposed method has three modules, which include the registration module, the verification module, and the authentication module. In the registration module, the user starts to register himself or herself using his/her voice. This happens via an application on the user's smartphone. Once the system, which resides in the cloud, receives the voice of the user, it then generates a secret key based on the user's identity. Next, the verification module starts by first making sure if the network, from which the user is trying to access the IoT device, is safe. The verification module must also to verify whether the user is a new user or a returning user. This verification is achieved based on the user voice submitted at the registration phase. If it is a new user, the system creates a new model for this user. If the identity claimer's voice is that of a legitimate returning user, the user will have access to the IoT network via the IoT gateway. In the authentication module, using the secret key generated in the registration module, the cloud-residing council encrypts the voice using a hash function and generates a session key with the IoT gateway, and then the encrypted voice is sent to the IoT device. The IoT device sends the encrypted voice to the IoT gateway to send the session key back to the IoT device that the user wants to connect to.

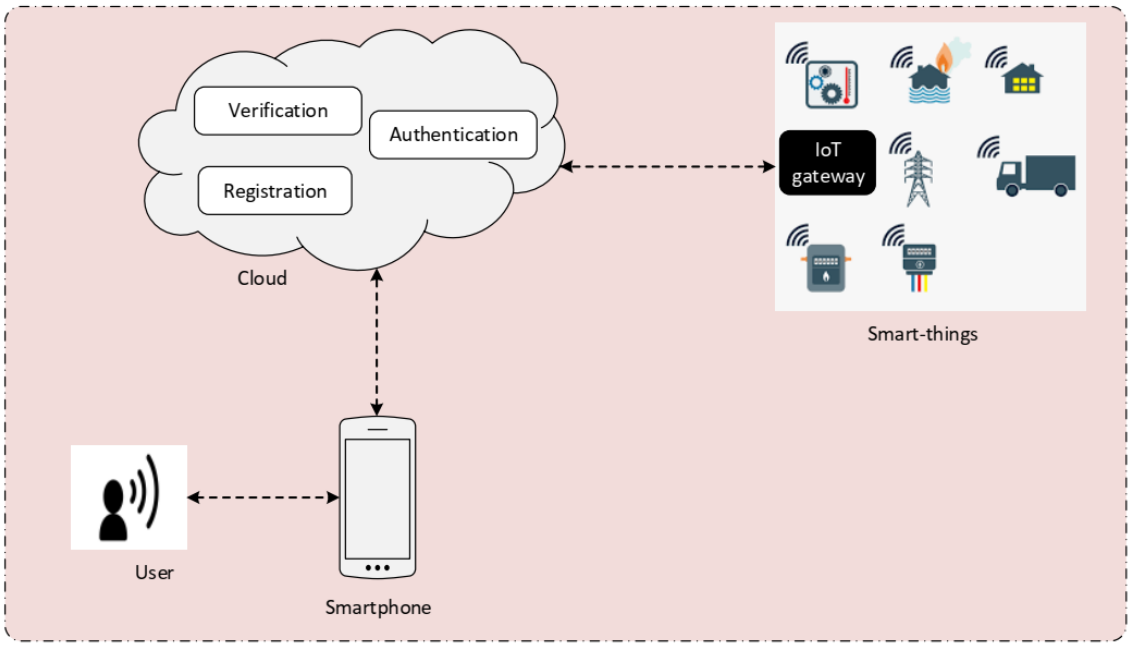

Figure 5. 1. Authentication model proposed for IoT user authentication

As shown in Figure 5.1, in the network scenario of the proposed authentication method, users can communicate with smart devices to access services. The network scenario contains a set of constrained devices. Each user is expected to use his smartphone to access IoT devices via an application connected to an IoT management council that resides in the cloud. Smart devices are accessed via the IoT gateway.

5.5 Deep Learning-Based Feature Extraction and Classifiers

An end-to-end deep learning neural network was used for feature extraction and classification in the proposed model. As a result of advances in audio devices, it has become increasingly difficult to determine typical characteristics of a replay speech based on conventional machine-learning methods. Thus, to detect basic characteristics in the replay speech, an end-to-end deep learning method was employed in this model. The deep learning method used in this model was comprised of CNN used at the frontend of the feature extraction, as well as at the backend classification of the model. In other words, the input features were first processed with the CNN layers to extract frame-level embedding. The CNN have residual blocks with identity mapping to facilitate the training of deep architecture. In addition, the CNN layer used in the model processes local adjacent time and frequency domains and repeated pooling operations are used to extract frame-level embedding. Sample architecture of the original CNN, as introduced by LeCun et al. (1969), is shown in Figure 5.2.

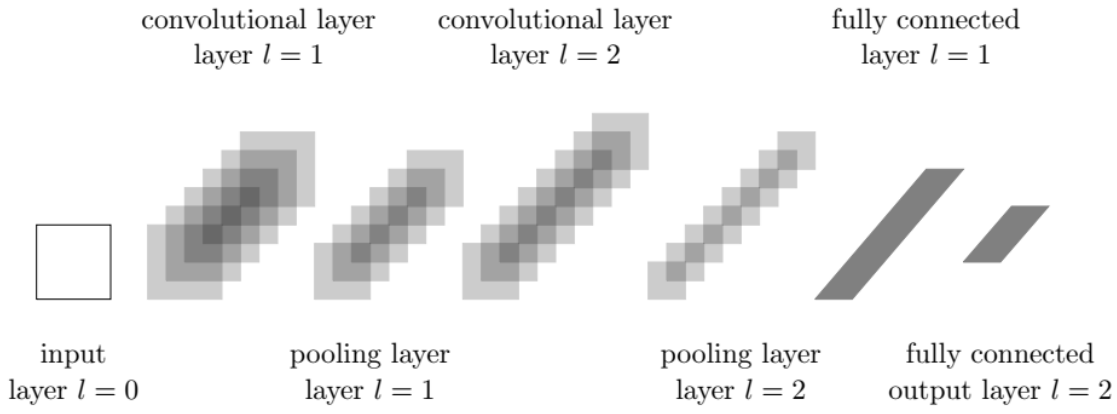

Figure 5. 2. Original architecture of CNN as proposed in [117]

Two disjoint sets of users were used in the training of the feature extraction and classifier. The development D set in the dataset was used for learning the CNN based extraction, while the classifiers were trained on the evaluation set E . A function $\phi(X)$ takes X user voice data of the development D dataset as input and returns a fixed feature vector. The learned function is then used to extract features for the voice data in the evaluation set E and train a classifier for each user.

Usually, it is reported, that for CNN architecture the input size of the data has to be specific for training and testing. This constraint is caused by the usage of fully connected layers at the end of the network architecture. That is to say, the convolution and pooling layers do not ask for a fixed size of input data. The problem between the pooling layer 2 and fully connected layer 1 is obvious in Figure 5.2. The last pooling layer *l2* is flattened to a vector dimensionality K and the fully connected layer *l1* uses a weight matrix of size $K + M$ where M is the output size of the fully connected layer. If CNN processes an input of a certain specific size, the output of the last pooling layer *l2* is expected to have a different size. That output size will not define the product of the fully connected layer *l1*. Hence, spatial pyramid pooling (SPP), as provided in [118], is used to obtain a fixed-sized representation for variable-size input data. In the training, the CNN is initialized with random weights and the training is performed with stochastic gradient descent to minimize the loss function. Mini batches of data are used to speed up the training. As such, the row data as presented in the dataset is used as CNN input. The probability of discriminative embedding x_i and x_j of the same speaker is modelled by their cosine Equation (1) as provided in [51].

$$cos\left(x_i,\ x_j\right) = x_i^T xj \qquad (1)$$

The triplet loss that takes in as input three samples as shown in Figure 5.3 including x^a , x^p , and x^n . That is to say, x^a is an utterance from a specific speaker (anchor), x^p is another utterance from the same speaker (positive), and x^n is an utterance from another speaker (negative). Updates are made to seek for the cosine similarity between the anchor and the positive examples are larger than the cosine similarity between the anchor and the negative examples. This is formally provided as

$$s_i^{x^p} - \alpha > s_i^{x^n} \qquad (2)$$

where, $s_i^{x^p}$ the cosine similarity between the anchor a and the positive example p for triplet i , and $s_i^{x^n}$ is the cosine similarity between the anchor a and the negative example n in triplet i . A minimum margin α is imposed between the similarities.

$$\therefore \qquad L = \sum_{i=0}^{N} [s_i^{x^n} - s_i^{x^p} + \alpha] \qquad (3)$$

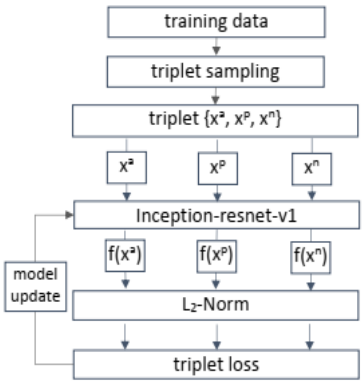

Figure 5. 3. Triplet loss of a speaker embedding training system

5.6 Dataset

The ASVspoof 2019 dataset was used for the experiments. The details of the dataset are given in Table 5.1. The dataset contains utterances of 107 speakers. The training set has 5400 genuine utterances and 48,600 spoofed utterances.

Table 5. 1 ASVspoof 2019 details

Sets	Total Speakers	Male	Female
Development	10	4	6
Training	20	8	12
Evaluation	48	21	27

5.7 Implementation and Results

This section demonstrates the implementation of the proposed IoT authentication model using a smartphone and a smartwatch. The management council of the application resides in the cloud. As for the proven prototype, a local private cloud was used. The user has an application in his/her smartphone to connect and authenticate him or herself with his/her smartwatch. The deep learning architecture was implemented using Keras with TensorFlow, an open-source neural network library written in Python to evaluate the results. The Keras library was selected for a number of reasons, but mainly because of its lightweight and ease of extensibility. Furthermore, Keras can process large amounts of data. The implementation consisted of pre-processing, creation of and training CNN feature extractor, and training the classifier, and testing stages. The two metrics used for the evaluation include accuracy and mean-squared error rate.

5.7.1 Pre-processing

In general, deep learning rarely takes raw audio signals, mainly because of the fluctuating scales in the signals. In this pre-processing, the scale problem of the raw audio data was solved by adopting the widely used pre-processing technique used in [119]. High-frequency signals were the centre of the pre-processing. Likewise, the adopted pre-processing method stabilized the scale of raw audio signals. The method is designed as $p(t) = s(t) - \alpha \cdot s(t-1)$, where $s(t)$ refers to the input signal at time t and α refers to the pre-emphasis coefficient, and $p(t)$ stands for pre-emphasized signals.

5.7.2 CNN Architecture

The main target of the project was to authenticate an IoT user in a lightweight manner. In the experiment, a user is using his smartphone to authenticate himself to his Android wearable device via an authentication system residing in the cloud. First column of the Table 5.2 represents the convolutional and residual network layers in the used architecture, while structure of the layers are given the in second column and the right most column provides the size of the pooling region or strides.

Table 5. 2 The CNN Network architecture

Layers	Structure	Stride
Conv1	5 x 5, 32	1 x 1
Rest32	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix} \times 3$	1 x 1
Conv2	5 x 5, 64	2 x 2
Rest64	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	3 x3
Conv3	5 x 5, 128	1 x 1
Rest128	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 3$	2 x 2
Conv4	5 x 5, 256	1 x 1
Rest256	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 3$	3 x 3
Conv5	5 x 5, 512	1 x 1
Rest512	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	2 x 2

The feature extraction happens in the user's smartphone. The deep learning approach-based feature extractor is used. The SPP used in this architecture where frame-level features were extracted by using the widely used 34-layer ResNet [51]. The architecture is detailed in Table 5.2. CNN architectures are

suitable for the reduction of spectral variations and modelling of spectral correlations of acoustic features. A distance-based loss function was used to discriminate between same-speaker and different-speaker utterance pairs. Also, at each iteration, candidate utterances were checked in the same mini batch. It provided faster training convergence.

5.7.3 Evaluation Result of the proposed Authentication Model

Only the training set of the dataset was selected to verify the efficiency of the model. The data consisted of 54,000 utterances with 5400 genuine utterances and 48,600 spoofed utterances. The evaluation metrics included classification performance, because it allowed us to predict which registered speaker corresponded to a given utterance. To evaluate the system performance, accuracy is the most important determinant; to accurately measure the ratio between the correct number of predictions and the total number of cases given as

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP stands for True Positive and TN refers to True Negative predictions. False Positive and False Negative prediction are given as FP and FN respectively. The lowest rate of 81.60% and highest rate of 92.56% of accuracies was observed. Accuracy alone cannot typically provide enough information to evaluate the prediction outcomes. That is to say, accuracy cannot necessarily reflect an indicator of a high classification performance. Accuracy cannot hold as much weight when a dataset with a huge amount of data is involved. To avoid this accuracy paradox, the loss function, or Mean Square Error rate (MSE), was used to pair with the accuracy calculations so that we could gain more insight into the evaluation of the model. This was treated as a regression problem with deep learning. Because the used dataset consists continuous set of data for each user, MSE is believed to be an appropriate metric. What the MSE achieved over the iterations is shown in Figure 5.4.

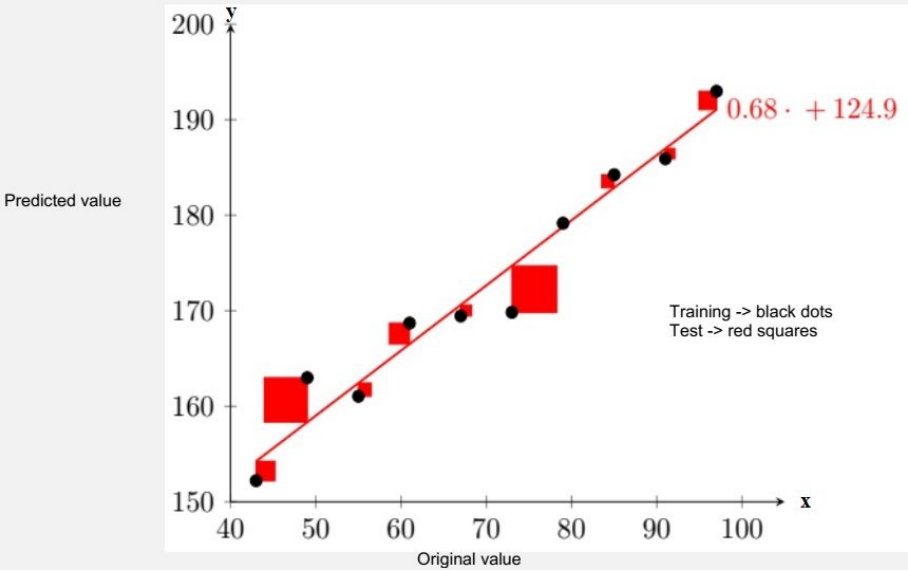

Figure 5. 4. Observed Mean Square Rate of the proposed authentication model

In addition to the authentication accuracy, the deep learning method was evaluated from different stances. In the first stance, the architecture was evaluated on the three sets of the ASV2019 dataset: development, training, and testing for text-independent recognition tasks. That is, the performance of a baseline i-vector system was compared to our approach. The i-vector system uses a frontend feature consisting of 12 MFCCs with a frame length of 30 ms that is feature is subject mean-normalized over a sliding window of up to 3 seconds. The baseline system uses 60-dimension feature vectors, created by Delta, and acceleration. Less frequency or unvoiced parts of the utterances are removed with energy-based activity detection. The UBM is a 1024-component full-covariance GMM and the system uses a 400-dimension i-vector extractor. In addition, before the PLDA scoring, i-vectors get centered and length normalized. The result is provided in terms of EER, where our system with the special pyramid pooling achieves 15.8% relative improvement over the baseline system.

5.8 Chapter Summary

In this work, a voice biometric-based authentication system for IoT systems has been proposed. A prototype application built on a private cloud was used for the evaluation of the system. The system used an end-to-end deep learning approach to extract features from a user's voice and subsequently classify them. The proposed system used ResNet CNN architecture that removes the fixed-size limitation, which improves the accuracy of the system in general. The CNN approach employed achieved state-of-the-art performance on the dataset of ASVspoof 2019. The Resnet CNN architecture coding is included in Appendix E.

Chapter 6: Conclusion and Future Works

6.1 Conclusion

The main objective for conducting this research was to develop a biometric-based secure lightweight authentication system that can be used for IoT devices. To achieve this objective, three research questions were developed and addressed throughout the process.

First, investigate a suitable biometric that can be used in IoT devices without the need for an additional hardware. To this end, we studied different biometric authentication techniques that are described in the literature. Accordingly, there are two main types, physical and behavioral. Physical biometrics can include fingerprints or iris scan, while behavioral biometrics can include keystroke or voiceprint. After comparing the various biometrics, the voiceprint was selected as the best candidate for authentication for ease of integration into the IoT environment; voiceprint does not necessitate dedicated hardware to be used in IoT. Consequently, the feasibility of adopting a lightweight voiceprint-based authentication mechanism in IoT was investigated. Experiments using the traditional open-source machine-learning tools revealed that the voice-based authentication system would be appropriate for adoption by IoT-based systems. The results of this investigation have been published [39, 72].

Second, a well known fact is that voiceprint-based authentication suffers from spoofing, in particular replay attacks. Therefore, to answer the second research question, a selection of features that can help detect a replay attack was investigated. A hybrid feature that combines MCFF and CQCC was proposed. The MFCC feature extraction is used for processing high frequency areas the speech utterance while the CQCC feature extraction is used for processing silent areas with low frequency. Moreover, MFCC is well-known in appropriately capturing voice features with high frequency and that the CQCC is a multi-resolution cepstral feature. Subsequently, the CNN networks were trained with the selected hybrid features. The CNN networks were used as the frontend together with a SVM backend combined into our voice-verification system (VVS). By comparing the proposed solution to a baseline (i.e., from [45]) that uses a “conventional” machine-learning frontend and backend, our results show that the SVSs successfully improved replay attack detection. The result obtained using the linear SVM was 7.9% and the result obtained using the kernel SVM was 7.1% as described in Section 3.6. In another experiment, DNN networks were trained with the selected hybrid features. The DNN networks were used as backend classifiers with GMM feature extractions. With the deep learning-based classifier, the results show improved replay attack detection as compared to traditional machine learning using baseline systems [45]. The results are shown in Table 4.2, with the Development dataset

we obtained 10.8%, with the Evaluation dataset we obtained 23.6%, and with the combination of Development and evaluation datasets we Obtained 10.8%.

Lastly, in response to our third question, a prototype of the voiceprint-based authentication system that can detect replay attacks was employed in an experiment to demonstrate its usability in IoT devices. The well-known CNN model, layer 34 ResNet, was used in the prototype. The experiment included a virtual environment that consisted of a smartphone, a smartwatch, and a cloud. The prototype was evaluated for its accuracy and precision. The results are very promising as discussed in Section 5.7.3.

In the validation of the answers made to the three questions coined to achieve the main objective of the study, we have used three datasets including MIT dataset, ASVspoof17 and ASVspoof19 datasets. For example, to answer to the first question of the study that involved studying the suitability of voice based biometric for the authentication in IoT environments, the MIT datasets have been use. The ASVspoof17 datasets have been used for the validation the CNN and DNN networks proposed for the replay attack detection (which answers to the second questions), while the ASVspoof19 datasets have been used for the validation of the ResNet based prototype that answers to the third question of the study.

6.1.1 Summary of implementation specifics

Throughout this work, the python program language was the main language employed for the deep learning models for two reasons. First, was the familiarity with python. The second reason is that python is recommended by the NIST. NIST compared various existing libraries used for speaker verification systems (SVSs). As a result, the python (namely, Keras and Tensorflow) neural network libraries were singled out to be of the highest integrity (i.e., estimated to have fewer errors). *In addition, the most recent studies that use deep learning models for SVSs use python.* This has great advantages in terms of reproducibility of our results. The source code of the ALIZE library, (Chapter 2) used for the feasibility study conducted to evaluate the suitability of voice biometrics-based authentication in IoT, is included in Appendix A for reproducibility. Likewise, the source code of the SVM machine used in the dissertation is shown in Appendix B so that it can be replicated for further experiments. The source code of the CNN architecture used for the data extraction in Chapter 3 is included in Appendix C for reproducibility. The code of the DNN architecture, used for classification of features in Chapter 4, is included in Appendix D for further examination. Finally, the source code of the Resnet architecture used in Chapter 5 is also provided in Appendix E for reproducibility as well.

The CNN networks are used to differentiate the genuine user voice from replay voice used by attackers to spoof the “true” identity of users. To accomplish this, CNN networks are expected to extract minor

discriminating characteristics that differentiate the genuine from the replayed voice. For example, the SVS model is first trained using the hybrid features extracted from the genuine voice. Subsequently, once the CNN frontend of the model has assimilated (i.e., trained) using features from the genuine voice, the model can correctly detect features extracted from replay voice with high confidence of up to 7.9% and 7.1% error rates when used with two backend classifiers, the linear SVM and EBF kernel respectively.

6.1.2 Summary of data used for validation

Very importantly, two datasets (ASVspoof 2017, ASVspoof 2019), both meant to characterize the pernicious replay type of spoofing attack, have been utilized to validate our proposed speaker verification approach which was demonstrated to improve on replay attack detection with a 15.8% Equal Error Rate (EER) .

For example, the ASVspoof 2017 dataset used in the DNN consisted of three subsets: 1) training, 2) development, and 3) evaluation *subsets*. In the DNN, the development and the evaluation subsets in the dataset were employed. The development set consisted of 760 genuine voices and 950 replay voice, while the evaluation set consisted of 1298 genuine voices and 12008 replay voices. All the genuine and replay voices in the two subsets were used for the model training and evaluation (i.e., testing). Hence, the n^{th} folder theory was not used in the experiment, because it was not applicable in the case of our SVS.

In the cases described in Chapters 3 and 4, accuracy is measured by the true/false positive and false/true negative rates of the replay attack detection. In Chapter 5, the accuracy is measured by the acceptance and rejection rates within the context of user authentication as discussed in Section 5.7.3. The improved accuracy of user authentication using our SVS provides a better approach compared to other recent studies [45, 55]. Thus, our authentication system has shown, using the aforementioned datasets an improved 15.8% relative accuracy.

6.2 Future IoT Authentication Research Directions

There is an abundance of security issues associated with the voice-based authentication systems such as voice conversion and mimicry attacks that have not been taken into consideration here and that are beyond the scope of this study. The authentication system provided in Chapter 5 has been evaluated for its resilience against replay spoof attack and has provided an accuracy of between 81.6% and 92.6%.

Developing better defenses against other spoofing attack methods and tactics is an open area for future research toward increasing IoT system resilience and creating generic machine learning defenses.

Furthermore, instantiating a real system using our proposed deep learning methods framework should be examined as an area of active research. Given the SVSs studied and validated, we would suggest that further development of a full-scale general-purpose IoT voice authentication system is feasible that can serve as one factor of a multifactor IoT authentication system.

References

1. Olazabal, O., et al. *Multimodal Biometrics for Enhanced IoT Security*. in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. 2019. IEEE.

2. Ren, Y., et al., *Replay attack detection based on distortion by loudspeaker for voice authentication.* Multimedia Tools and Applications, 2019. **78**(7): p. 8383-8396.

3. Gupta, S. and S. Chatterjee, *Text dependent voice based biometric authentication system using spectrum analysis and image acquisition*, in *Advances in Computer Science, Engineering & Applications*. 2012, Springer. p. 61-70.

4. Kolkata, C., *About Voice biometric and Speaker Recognition*. 2014.

5. Thakur, A.S. and N. Sahayam, *Speech recognition using euclidean distance.* International Journal of Emerging Technology and Advanced Engineering, 2013. **3**(3): p. 587-590.

6. Petrovska-Delacrétaz, D., A. El Hannani, and G. Chollet, *Text-independent speaker verification: state of the art and challenges*, in *Progress in nonlinear speech processing*. 2007, Springer. p. 135-169.

7. Rosenberg, A.E., F. Bimbot, and S. Parthasarathy, *Overview of speaker recognition*, in *Springer Handbook of Speech Processing*. 2008, Springer. p. 725-742.

8. Nainan, S. and V. Kulkarni. *Performance evaluation of text independent automatic speaker recognition using VQ and GMM*. in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*. 2016. ACM.

9. McLaren, M., *Automatic Speaker Recognition for Authenticating Users in the Internet of Things*. August 26, 2016.

10. Sharma, G. and S. Kalra, *A Lightweight User Authentication Scheme for Cloud-IoT Based Healthcare Services.* Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 2019. **43**(1): p. 619-636.

11. El-hajj, M., et al., *A survey of internet of things (IoT) Authentication schemes.* Sensors, 2019. **19**(5): p. 1141.

12. Dhillon, P.K. and S. Kalra, *A lightweight biometrics based remote user authentication scheme for IoT services.* Journal of Information Security and Applications, 2017. **34**: p. 255-270.

13. Yassine, A., et al., *IoT big data analytics for smart homes with fog and cloud computing.* Future Generation Computer Systems, 2019. **91**: p. 563-573.

14. Ferrag, M.A., L. Maglaras, and A. Derhab, *Authentication and Authorization for Mobile IoT Devices Using Biofeatures: Recent Advances and Future Trends.* Security and Communication Networks, 2019. **2019**.

15. Hamidi, H., *An approach to develop the smart health using Internet of Things and authentication based on biometric technology.* Future generation computer systems, 2019. **91**: p. 434-449.

16. Moussa, A.N., N. Ithnin, and A. Zainal, *CFaaS: bilaterally agreed evidence collection.* Journal of Cloud Computing, 2018. **7**(1): p. 1.

17. Thullier, F., B. Bouchard, and B.-A. Menelas, *A Text-Independent Speaker Authentication System for Mobile Devices.* Cryptography, 2017. **1**(3): p. 16.

18. Zhang, X., et al. *Voice Biometric Identity Authentication System Based on Android Smart Phone*. in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. 2018. IEEE.

19. Khitrov, A. and K. Simonchik, *System for text-dependent speaker recognition and method thereof*. 2019, Google Patents.

20. Moussa, A.N., et al. *A Consumer-Oriented Cloud Forensic Process Model*. in *2019 IEEE 10th Control and System Graduate Research Colloquium (ICSGRC)*. 2019. IEEE.

21. El Hannani, A., et al., *Text-independent speaker verification*, in *Guide to Biometric Reference Systems and Performance Evaluation*. 2009, Springer. p. 167-211.

22. Rosenberg, A.E. and S. Parthasarathy. *Speaker background models for connected digit password speaker verification*. in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. 1996. IEEE.

23. Oak, R., *A Literature Survey on Authentication Using Behavioural Biometric Techniques*, in *Intelligent Computing and Information and Communication*. 2018, Springer. p. 173-181.

24. Moussa, A.N., N.B. Ithnin, and O.A. Miaikil. *Conceptual forensic readiness framework for infrastructure as a service consumers*. in *2014 IEEE Conference on Systems, Process and Control (ICSPC 2014)*. 2014. IEEE.

25. Khamis, M., et al. *Gazetouchpass: Multimodal authentication using gaze and touch on mobile devices*. in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2016.

26. Arteaga-Falconi, J.S., H. Al Osman, and A. El Saddik, *ECG authentication for mobile devices.* IEEE Transactions on Instrumentation and Measurement, 2015. **65**(3): p. 591-600.

27. Shahzad, M., A.X. Liu, and A. Samuel, *Behavior based human authentication on touch screen devices using gestures and signatures.* IEEE Transactions on Mobile Computing, 2016. **16**(10): p. 2726-2741.

28. Ali, Z., et al., *Edge-centric multimodal authentication system using encrypted biometric templates.* Future Generation Computer Systems, 2018. **85**: p. 76-87.

29. Frank, M., et al., *Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication.* IEEE transactions on information forensics and security, 2012. **8**(1): p. 136-148.

30. Tasia, C.J., et al., *Two novel biometric features in keystroke dynamics authentication systems for touch screen devices.* Security and Communication Networks, 2014. **7**(4): p. 750-758.

31. Khan, M.K., S. Kumari, and M.K. Gupta, *More efficient key-hash based fingerprint remote authentication scheme using mobile device.* Computing, 2014. **96**(9): p. 793-816.

32. Mahbub, U., et al. *Partial face detection for continuous authentication*. in *2016 IEEE International Conference on Image Processing (ICIP)*. 2016. IEEE.

33. De Marsico, M., et al., *Firme: Face and iris recognition for mobile engagement.* Image and Vision Computing, 2014. **32**(12): p. 1161-1172.

34. Shin, D.-G. and M.-S. Jun. *Home IoT device certification through speaker recognition*. in *2015 17th International Conference on Advanced Communication Technology (ICACT)*. 2015. IEEE.

35. Atwady, Y. and M. Hammoudeh. *A survey on authentication techniques for the internet of things*. in *Proceedings of the International Conference on Future Networks and Distributed Systems*. 2017. ACM.

36. Sahoo, T.R. and S. Patra, *Silence Removal and Endpoint Detection of Speech Signal for Text Independent Speaker Identification.* International Journal of Image, Graphics & Signal Processing, 2014. **6**(6).

37. Dhakal, P., et al., *A Near Real-Time Automatic Speaker Recognition Architecture for Voice-Based User Interface.* Machine Learning and Knowledge Extraction, 2019. **1**(1): p. 504-520.

38. Guillaume, G., *SPro: speech signal processing toolkit.* Software available at http://gforge. inria. fr/projects/spro, 2004.

39. Salahaldeen Duraibi, F.T.S.a.W.A., *"Voice Biometric Identity Authentication Model for IoT Devices",.* International Journal of Security, Privacy and Trust Management (IJSPTM), , 2020. **Vol. 9,** (No.2).

40. Wang, X., et al., *ASVspoof 2019: a large-scale public database of synthetized, converted and replayed speech.* Comput. Speech Lang., vol. 64, 2020.

41. Reynolds, D.A., T.F. Quatieri, and R.B. Dunn, *Speaker verification using adapted Gaussian mixture models.* Digital signal processing, 2000. **10**(1-3): p. 19-41.

42. Lukic, Y., et al. *Speaker identification and clustering using convolutional neural networks*. in *2016 IEEE 26th international workshop on machine learning for signal processing (MLSP)*. 2016. IEEE.

43. Irum, A. and A. Salman, *Speaker Verification Using Deep Neural Networks: A.* International Journal of Machine Learning and Computing, 2019. **9**(1).

44. Nagrani, A., et al., *Voxceleb: Large-scale speaker verification in the wild.* Computer Speech & Language, 2020. **60**: p. 101027.

45. Dey, S., et al. *Deep neural network based posteriors for text-dependent speaker verification*. in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016. IEEE.

46. Salehghaffari, H., *Speaker verification using convolutional neural networks.* arXiv preprint arXiv:1803.05427, 2018.

47. Kanagasundaram, A., *Speaker verification using I-vector features*. 2014, Queensland University of Technology.

48. Richardson, F., D. Reynolds, and N. Dehak, *A unified deep neural network for speaker and language recognition.* arXiv preprint arXiv:1504.00923, 2015.

49. Snyder, D., et al. *Deep neural network-based speaker embeddings for end-to-end speaker verification*. in *2016 IEEE Spoken Language Technology Workshop (SLT)*. 2016. IEEE.

50. Matějka, P., et al. *Analysis of DNN approaches to speaker identification*. in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. 2016. IEEE.

51. Li, C., et al., *Deep speaker: an end-to-end neural speaker embedding system.* arXiv preprint arXiv:1705.02304, 2017. **650**.

52. Snyder, D., et al. *Deep Neural Network Embeddings for Text-Independent Speaker Verification*. in *Interspeech*. 2017.

53. Abdel-Hamid, O., L. Deng, and D. Yu. *Exploring convolutional neural network structures and optimization techniques for speech recognition*. in *Interspeech*. 2013.

54. McLaren, M., et al. *Application of convolutional neural networks to speaker recognition in noisy conditions*. in *Fifteenth Annual Conference of the International Speech Communication Association*. 2014.

55. Chen, K. and A. Salman, *Learning speaker-specific characteristics with a deep neural architecture.* IEEE Transactions on Neural Networks, 2011. **22**(11): p. 1744-1756.

56. Wang, M., et al. *Speaker recognition using convolutional neural network with minimal training data for smart home solutions*. in *2018 11th International Conference on Human System Interaction (HSI)*. 2018. IEEE.

57. Huang, L., Y. Gan, and H. Ye. *Audio-replay Attacks Spoofing Detection for Automatic Speaker Verification System*. in *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. 2019. IEEE.

58. Huang, L. and C.-M. Pun. *Audio replay spoof attack detection using segment-based hybrid feature and densenet-LSTM network*. in *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. 2019. IEEE.

59. Lavrentyeva, G., et al. *Audio replay attack detection with deep learning frameworks*. in *Interspeech*. 2017.

60. Nagarsheth, P., et al. *Replay Attack Detection Using DNN for Channel Discrimination*. in *Interspeech*. 2017.

61. Singh, M., J. Mishra, and D. Pati. *Replay attack: Its effect on GMM-UBM based text-independent speaker verification system*. in *2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON)*. 2016. IEEE.

62. Chatfield, K., et al., *Return of the devil in the details: Delving deep into convolutional nets.* arXiv preprint arXiv:1405.3531, 2014.

63. Glorot, X. and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*. in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

64. Lee, H., et al. *Unsupervised feature learning for audio classification using convolutional deep belief networks*. in *Advances in neural information processing systems*. 2009.

65. Dieleman, S., et al., *Lasagne: first release.* Zenodo: Geneva, Switzerland, 2015. **3**.

66. Milton, A., S.S. Roy, and S.T. Selvi, *SVM scheme for speech emotion recognition using MFCC feature.* International Journal of Computer Applications, 2013. **69**(9).

67. Cortes, C. and V. Vapnik, *Support-vector networks.* Machine learning, 1995. **20**(3): p. 273-297.

68. Kinnunen, T., et al. *Reddots replayed: A new replay spoofing attack corpus for text-dependent speaker verification research*. in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. IEEE.

69. Evans, N., et al., *Speaker recognition anti-spoofing*, in *Handbook of biometric anti-spoofing*. 2014, Springer. p. 125-146.

70.Jain, A.K., R. Bolle, and S. Pankanti, *Biometrics: personal identification in networked society*. Vol. 479. 2006: Springer Science & Business Media.

71. Gayathri, M., C. Malathy, and M. Prabhakaran. *A Review on Various Biometric Techniques, Its Features, Methods, Security Issues and Application Areas*. in *International Conference On Computational Vision and Bio Inspired Computing*. 2019. Springer.

72. Duraibi, S., F.T. Sheldon, and W. Alhamdani, "*VOICE BIOMETRIC IDENTITY AUTHENTICATION MODEL FOR IOT DEVICES"* International Jouranl of security, privacy and Turst Managemnet(IJSPTM), Vol. 9, No.2.

73. Wu, Z., et al., *Spoofing and countermeasures for speaker verification: A survey.* speech communication, 2015. **66**: p. 130-153.

74. Vestman, V., et al., *Voice mimicry attacks assisted by automatic speaker verification.* Computer Speech & Language, 2020. **59**: p. 36-54.

75. Pal, M. and G. Saha, *On robustness of speech based biometric systems against voice conversion attack.* Applied Soft Computing, 2015. **30**: p. 214-228.

76. Alegre, F., A. Janicki, and N. Evans. *Re-assessing the threat of replay spoofing attacks against automatic speaker verification*. in *2014 International Conference of the Biometrics Special Interest Group (BIOSIG)*. 2014. IEEE.

77. Hong, Q., S. Wang, and Z. Liu. *A robust speaker-adaptive and text-prompted speaker verification system*. in *Chinese Conference on Biometric Recognition*. 2014. Springer.

78. De Leon, P.L., et al., *Evaluation of speaker verification security and detection of HMM-based synthetic speech.* IEEE Transactions on Audio, Speech, and Language Processing, 2012. **20**(8): p. 2280-2290.

79. Bredin, H., et al. *Detecting replay attacks in audiovisual identity verification*. in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. 2006. IEEE.

80. Nematollahi, M., et al. *Digital speech watermarking for anti-spoofing attack in speaker recognition*. in *2014 IEEE REGION 10 SYMPOSIUM*. 2014. IEEE.

81. Shang, W. and M. Stevenson, *Detection of speech playback attacks using robust harmonic trajectories.* Computer Speech & Language. **65**: p. 101133.

82. Rafi B, S. and S.R.M. Kodukula, *Importance of analytic phase of the speech signal for detecting Replay attacks in automatic speaker verification systems.* 2019.

83. Shang, W. and M. Stevenson. *A playback attack detector for speaker verification systems*. in *2008 3rd International Symposium on Communications, Control and Signal Processing*. 2008. IEEE.

84. Faundez-Zanuy, M., M. Hagmüller, and G. Kubin, *Speaker verification security improvement by means of speech watermarking.* Speech communication, 2006. **48**(12): p. 1608-1619.

85. Villalba, J. and E. Lleida. *Preventing replay attacks on speaker verification systems*. in *2011 Carnahan Conference on Security Technology*. 2011. IEEE.

86. Korshunov, P., et al. *Overview of BTAS 2016 speaker anti-spoofing competition*. in *2016 IEEE 8th international conference on biometrics theory, applications and systems (BTAS)*. 2016. IEEE.

87. Shaikh, R. and M. Sasikumar, *Data Classification for achieving Security in cloud computing.* Procedia computer science, 2015. **45**: p. 493-498.

88. Zhang, C., C. Yu, and J.H. Hansen, *An investigation of deep-learning frameworks for speaker verification antispoofing.* IEEE Journal of Selected Topics in Signal Processing, 2017. **11**(4): p. 684-694.

89. Xu, J., et al., *Deep multi-metric learning for text-independent speaker verification.* Neurocomputing, 2020. **410**: p. 394-400.

90. Ustinova, E., Y. Ganin, and V. Lempitsky. *Multi-region bilinear convolutional neural networks for person re-identification*. in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017. IEEE.

91. Hafemann, L.G., R. Sabourin, and L.S. Oliveira. *Writer-independent feature learning for offline signature verification using deep convolutional neural networks*. in *2016 international joint conference on neural networks (IJCNN)*. 2016. IEEE.

92. Li, L., et al., *A study on replay attack and anti-spoofing for automatic speaker verification.* arXiv preprint arXiv:1706.02101, 2017.

93. Bakar, B. and C. Hanilçi. *An Experimental Study on Audio Replay Attack Detection Using Deep Neural Networks*. in *2018 IEEE Spoken Language Technology Workshop (SLT)*. 2018. IEEE.

94. Gomez-Alanis, A., et al., *A gated recurrent convolutional neural network for robust spoofing detection.* IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2019. **27**(12): p. 1985-1999.

95. Kinnunen, T., et al., *The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection.* 2017. in Proc. INTERSPEECH, 2017, pp. 2– 6.

96. Dai, H.-N., et al., *Big data analytics for manufacturing internet of things: opportunities, challenges and enabling technologies.* Enterprise Information Systems, 2019: p. 1-25.

97. Adhikary, T., et al. *The Internet of Things (IoT) Augmentation in Healthcare: An Application Analytics*. in *International Conference on Intelligent Computing and Communication Technologies*. 2019. Springer.

98. Arasteh, H., et al. *Iot-based smart cities: a survey*. in *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*. 2016. IEEE.

99. Chunli, L. *Intelligent transportation based on the Internet of Things*. in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*. 2012. IEEE.

100. Zhang, Y., et al., *A consistency-guaranteed approach for Internet of Things software refactoring.* International Journal of Distributed Sensor Networks, 2020. **16**(1): p. 1550147720901680.

101. Lang, J.P., et al., *Voice control of a media playback system*. 2019, Google Patents.

102. Kagita, M.K. *Security and Privacy Issues for Business Intelligence in lOT*. in *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*. 2019. IEEE.

103. Hernández-Ramos, J.L., et al., *SAFIR: Secure access framework for IoT-enabled services on smart buildings.* Journal of Computer and System Sciences, 2015. **81**(8): p. 1452-1463.

104. Micro, T., *Are your wearables fit to secure you? researchers outline 3 attack surfaces*. 2018, March.

105. Mail, D., *Do YOU suffer from password rage? A third of people have thrown a tantrum after forgetting login details*. 2015.

106. Meng, Y., et al. *Wivo: Enhancing the security of voice control system via wireless signal in iot environment*. in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 2018.

107. Ergünay, S.K., et al. *On the vulnerability of speaker verification to realistic voice spoofing*. in *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. 2015. IEEE.

108. Boles, A. and P. Rad. *Voice biometrics: Deep learning-based voiceprint authentication system*. in *2017 12th System of Systems Engineering Conference (SoSE)*. 2017. IEEE.

109. Alhussein, M. and G. Muhammad, *Voice pathology detection using deep learning on mobile healthcare framework.* IEEE Access, 2018. **6**: p. 41034-41041.

110. Bazrafkan, S. and P. Corcoran. *Enhancing iris authentication on handheld devices using deep learning derived segmentation techniques*. in *2018 IEEE international conference on consumer electronics (ICCE)*. 2018. IEEE.

111. Rattani, A., N. Reddy, and R. Derakhshani. *Multi-biometric convolutional neural networks for mobile user authentication*. in *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*. 2018. IEEE.

112. Traore, I., et al. *Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments*. in *2012 fourth international conference on digital home*. 2012. IEEE.

113. Das, R., et al. *A deep learning approach to IoT authentication*. in *2018 IEEE International Conference on Communications (ICC)*. 2018. IEEE.

114. Bayar, B. and M.C. Stamm. *A deep learning approach to universal image manipulation detection using a new convolutional layer*. in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*. 2016.

115. Nandy, T., et al., *Review on security of Internet of Things authentication mechanism.* IEEE Access, 2019. **7**: p. 151054-151089.

116. Hassan, W.H., *Current research on Internet of Things (IoT) security: A survey.* Computer networks, 2019. **148**: p. 283-294.

117. LeCun, Y., et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.

118. He, K., et al., *Spatial pyramid pooling in deep convolutional networks for visual recognition.* IEEE transactions on pattern analysis and machine intelligence, 2015. **37**(9): p. 1904-1916.

119. Hannun, A., et al., *Deep speech: Scaling up end-to-end speech recognition.* arXiv preprint arXiv:1412.5567, 2014.

Appendix A - Alize code

In studying the feasibility of adopting the voice biometric based authentication system for IoT devices described in Chapter 2, the Alize library has been used to simulate the base of the voice verification system. The source code of the system used for the implementation of the authentication model in IoT is as follows:

```
package
fr.univavignon.alize.AndroidALIZEDemo;

                import android.graphics.Color;
                import android.os.Bundle;
                import android.view.View;
                import android.widget.TextView;
                import AlizeSpkRec.AlizeException;
                import AlizeSpkRec.SimpleSpkDetSystem;
                /**
                 * Activity meant to identify or verify a speaker.
                 * Identify is used when the system try to recognize a
                speaker in the system with the current record.
                 * Verify is used when the system is listening to the
                current record and determine if it's the right speaker
                or not.
                 *
                 * @author Nicolas Duret
                 */
                public class VerificationActivity extends RecordActivity
                {
                    private final int ERROR_COLOR = Color.RED;
                    private final int SUCCESS_COLOR =
                Color.rgb(0,150,0);
                    private String speakerId, speakerName, activityName;
                    private TextView resultText;
                    private boolean identify = false;
                    @Override
                    public void onCreate(Bundle savedInstanceState) {
                        try {
                            super.onCreate(savedInstanceState);
                            setContentView(R.layout.verification);
                            speakerId =
                getIntent().getStringExtra("speakerId");
                            speakerName =
                demoSpkRecSystem.getSpeakerName(speakerId);
                            resultText = findViewById(R.id.result_text);
```

```
            startRecordButton =
findViewById(R.id.startBtn);
            stopRecordButton =
findViewById(R.id.stopBtn);
            timeText = findViewById(R.id.timeText);
            activityName =
getString(R.string.verification);
            String title =
getResources().getString(R.string.verify_activity_title,
speakerName);
            if (speakerName.isEmpty()) {
                identify = true;
                activityName =
getString(R.string.identify);
                title =
getResources().getString(R.string.identify_speaker);
            }
            setTitle(title);
            startRecordButton.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    resultText.setText("");
                    startRecording();
                }
            });
            stopRecordButton.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    stopRecording();
                }
            });
        }
        catch (Throwable e) {
            e.printStackTrace();
        }
    }
    protected void afterRecordProcessing() {
        String result = "";
        resultText.setTextColor(ERROR_COLOR);
        if (recordError) {
            resultText.setText(result);

makeToast(getResources().getString(R.string.recording_no
t_completed));
```

```
            return;
        }
        if (identify) {
            //Try to match a speaker with the record
            try {
                //This is the other main task of speaker
recognition: compare a recording with all the speakers
                //known to the system to determine whose
voice it is (or reject it as unknown).
                SimpleSpkDetSystem.SpkRecResult
identificationResult =
demoSpkRecSystem.identifySpeaker();
                if (identificationResult.match) {
                    result =
getString(R.string.identify_match_text) + "\n" +
demoSpkRecSystem.getSpeakerName(identificationResult.spe
akerId) + "\n" + getString(R.string.match_score) + "\n"
+ identificationResult.score;

resultText.setTextColor(SUCCESS_COLOR);
                }
                else {
                    result =
getString(R.string.identify_no_match_text) + "\n" +
getString(R.string.match_score) + "\n" +
identificationResult.score;

resultText.setTextColor(ERROR_COLOR);
                }
            }
            catch (AlizeException e) {
                e.printStackTrace();
                result =
getString(R.string.verification_failed_not_enough_featur
es);
            } catch (Throwable e) { //TODO catch proper
exception
                e.printStackTrace();
                result =
getString(R.string.verification_failed_not_enough_featur
es);
            }
        }
        else {
            //Compare the record with the speaker model
            try {
```

```
                //Compare the audio signal with the
speaker model we created earlier.
                SimpleSpkDetSystem.SpkRecResult
verificationResult =
demoSpkRecSystem.verifySpeaker(speakerId);
                if (verificationResult.match) {
                    result =
getString(R.string.verify_match_text) + "\n" +
getString(R.string.match_score) + "\n" +
verificationResult.score;
resultText.setTextColor(SUCCESS_COLOR);
}
else {
result = getString(R.string.verify_no_match_text,
speakerName) + "\n" + getString(R.string.match_score) +
"\n" + verificationResult.score;
resultText.setTextColor(ERROR_COLOR);
}
}
catch (AlizeException e) {
e.printStackTrace();
result =
getString(R.string.verification_failed_not_enough_featur
es);
} catch (Throwable e) { //TODO catch proper exception
e.printStackTrace();
result =
getString(R.string.verification_failed_not_enough_featur
es);
}
}
resultText.setText(result);
try {
//Reset input, since we will not make any more use of
this audio signal.
demoSpkRecSystem.resetAudio();       //Reset the audio
samples of the Alize system.
demoSpkRecSystem.resetFeatures();    //Reset the
features of the Alize system.
} catch (AlizeException e) { e.printStackTrace();
}}}
```

Appendix B - CNN

The Convolutional neural network (CNN) has been employed in the Chapter 3 in order to use it as feature extraction. In this appendix the coding of the python programming used the TensorFlow is provided.

```
Import OS
       from tqdm import tqdm
       import pandas as pd
       import numpy as np
       import matplotlib.pyplot as plt
       from scipy.io import wavfile
       from python_speech_features import mfcc, logfbank
       import librosa
       def plot_signals(signals):
           fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,
                                    sharey=True, figsize=(20,5))
           fig.suptitle('Time Series', size=16)
           i = 0
           for x in range(2):
               for y in range(5):
                   axes[x,y].set_title(list(signals.keys())[i])
                   axes[x,y].plot(list(signals.values())[i])
                   axes[x,y].get_xaxis().set_visible(False)
                   axes[x,y].get_yaxis().set_visible(False)
                   i += 1
       def plot_fft(fft):
           fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,
                                    sharey=True, figsize=(20,5))
           fig.suptitle('Fourier Transforms', size=16)
           i = 0
           for x in range(2):
               for y in range(5):
                   data = list(fft.values())[i]
                   Y, freq = data[0], data[1]
                   axes[x,y].set_title(list(fft.keys())[i])
                   axes[x,y].plot(freq, Y)
                   axes[x,y].get_xaxis().set_visible(False)
                   axes[x,y].get_yaxis().set_visible(False)
                   i +=1
       def plot_fbank(fbank):
           fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,
                                    sharey=True, figsize=(20,5))
           fig.suptitle('Filter Bank Coefficients', size=16)
           i = 0
           for x in range(2):
```

```
        for y in range(5):
            axes[x,y].set_title(list(fbank.keys())[i])
            axes[x,y].imshow(list(fbank.values())[i],
                    cmap='hot', interpolation='nearest')
            axes[x,y].get_xaxis().set_visible(False)
            axes[x,y].get_yaxis().set_visible(False)
            i += 1
def plot_mfccs(mfccs):
    fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,
                             sharey=True, figsize=(20,5))
    fig.suptitle('Mel Frequency Cepstrum Coefficients', size=16)
    i = 0
    for x in range(2):
        for y in range(5):
            axes[x,y].set_title(list(mfccs.keys())[i])
            axes[x,y].imshow(list(mfccs.values())[i],
                    cmap='hot', interpolation='nearest')
            axes[x,y].get_xaxis().set_visible(False)
            axes[x,y].get_yaxis().set_visible(False)
            i += 1
def envelope(y,rate,threshold):
    mask =[]
    y=pd.Series(y).apply(np.abs)

y_mean=y.rolling(window=int(rate/10),min_periods=1,center=True).m
ean()
    for mean in y_mean:
        if mean > threshold:
            mask.append(True)
        else:
            mask.append(False)
    return mask
def calc_fft(y,rate):
    n=len(y)
    freq=np.fft.rfftfreq(n,d=1/rate)
    Y=abs(np.fft.rfft(y)/n)
    return(Y,freq)
df = pd.read_csv('instruments.csv')
df.set_index('fname',inplace=True)
for i in df.index:
    rate,signal = wavfile.read('wavfiles/'+i)
    df.at[i,'length'] = signal.shape[0]/rate
classes=list(np.unique(df.label))
class_dist = df.groupby(['label'])['length'].mean()
fig,ax=plt.subplots()
ax.set_title('Class Distribution',y=1.08)
```

```
ax.pie(class_dist,labels=class_dist.index,autopct='%1.1f%%',shado
w=False,startangle=90)
ax.axis('equal')
plt.show()
df.reset_index(inplace=True)
signals={}
fft={}
fbank={}
mfccs={}
for c in classes:
    wav_file=df[df.label ==c].iloc[0,0]
    signal,rate=librosa.load('wavfiles/'+wav_file,sr=44100)
    mask = envelope(signal,rate,0.0005)
    signal=signal[mask]
    signals[c] = signal
    fft[c]=calc_fft(signal,rate)
    bank=logfbank(signal[:rate],rate,nfilt=16,nfft=1103).T
    fbank[c] =bank
    mel =mfcc(signal[:rate],rate,numcep = 13,nfilt=26,nfft=1103)
    mfccs[c]=mel
plot_signals(signals)
plt.show()
plot_fft(fft)
plt.show()
plot_fbank(fbank)
plt.show()
plot_mfccs(mfccs)
plt.show()
if len(os.listdir('clean')) == 0:
    for f in tqdm(df.fname):
        signal, rate=librosa.load('wavfiles/'+f,sr=16000)
        mask = envelope(signal,rate,0.0005)

wavfile.write(filename='clean/'+f,rate=rate,data=signal[mask])
```

Appendix C - SVM

The coding of the support vector machine (SVM) used for the feature classification in the speaker verification model discussed in Chapter 3. As explained previously, python programming language has been used in this research. Hence, the coding of the SVM employed in Chapter 3 is as follows:

```
Import  numpy as np
        import pandas as pd
        import statsmodels.api as sm
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.model_selection import train_test_split as tts
        from sklearn.metrics import accuracy_score, recall_score,
        precision_score
        from sklearn.utils import shuffle
        # >> FEATURE SELECTION << #
        def remove_correlated_features(X):
            corr_threshold = 0.9
            corr = X.corr()
            drop_columns = np.full(corr.shape[0], False, dtype=bool)
            for i in range(corr.shape[0]):
                for j in range(i + 1, corr.shape[0]):
                    if corr.iloc[i, j] >= corr_threshold:
                        drop_columns[j] = True
            columns_dropped = X.columns[drop_columns]
            X.drop(columns_dropped, axis=1, inplace=True)
            return columns_dropped
        def remove_less_significant_features(X, Y):
            sl = 0.05
            regression_ols = None
            columns_dropped = np.array([])
            for itr in range(0, len(X.columns)):
                regression_ols = sm.OLS(Y, X).fit()
                max_col = regression_ols.pvalues.idxmax()
                max_val = regression_ols.pvalues.max()
                if max_val > sl:
                    X.drop(max_col, axis='columns', inplace=True)
                    columns_dropped = np.append(columns_dropped,
        [max_col])
                else:
                    break
            regression_ols.summary()
            return columns_dropped
        ##############################
        # >> MODEL TRAINING << #
```

```
def compute_cost(W, X, Y):
    # calculate hinge loss
    N = X.shape[0]
    distances = 1 - Y * (np.dot(X, W))
    distances[distances < 0] = 0  # equivalent to max(0,
distance)
    hinge_loss = regularization_strength * (np.sum(distances) /
N)
    # calculate cost
    cost = 1 / 2 * np.dot(W, W) + hinge_loss
    return cost
# I haven't tested it but this same function should work for
# vanilla and mini-batch gradient descent as well
def calculate_cost_gradient(W, X_batch, Y_batch):
    # if only one example is passed (eg. in case of SGD)
    if type(Y_batch) == np.float64:
        Y_batch = np.array([Y_batch])
        X_batch = np.array([X_batch])  # gives multidimensional
array
    distance = 1 - (Y_batch * np.dot(X_batch, W))
    dw = np.zeros(len(W))
    for ind, d in enumerate(distance):
        if max(0, d) == 0:
            di = W
        else:
            di = W - (regularization_strength * Y_batch[ind] *
X_batch[ind])
        dw += di
    dw = dw/len(Y_batch)  # average
    return dw
def sgd(features, outputs):
    max_epochs = 5000
    weights = np.zeros(features.shape[1])
    nth = 0
    prev_cost = float("inf")
    cost_threshold = 0.01  # in percent
    # stochastic gradient descent
    for epoch in range(1, max_epochs):
        # shuffle to prevent repeating update cycles
        X, Y = shuffle(features, outputs)
        for ind, x in enumerate(X):
            ascent = calculate_cost_gradient(weights, x, Y[ind])
            weights = weights - (learning_rate * ascent)
        # convergence check on 2^nth epoch
        if epoch == 2 ** nth or epoch == max_epochs - 1:
            cost = compute_cost(weights, features, outputs)
```

```
            print("Epoch is: {} and Cost is: {}".format(epoch,
cost))
            # stoppage criterion
            if abs(prev_cost - cost) < cost_threshold *
prev_cost:
                return weights
            prev_cost = cost
            nth += 1
    return weights
########################
def init():
    print("reading dataset...")
    # read data in pandas (pd) data frame
    data = pd.read_csv('./data/data.csv')
    # drop last column (extra column added by pd)
    # and unnecessary first column (id)
    data.drop(data.columns[[-1, 0]], axis=1, inplace=True)
    print("applying feature engineering...")
    # convert categorical labels to numbers
    diag_map = {'M': 1.0, 'B': -1.0}
    data['diagnosis'] = data['diagnosis'].map(diag_map)
    # put features & outputs in different data frames
    Y = data.loc[:, 'diagnosis']
    X = data.iloc[:, 1:]
    # filter features
    remove_correlated_features(X)
    remove_less_significant_features(X, Y)
    # normalize data for better convergence and to prevent
overflow
    X_normalized = MinMaxScaler().fit_transform(X.values)
    X = pd.DataFrame(X_normalized)
    # insert 1 in every row for intercept b
    X.insert(loc=len(X.columns), column='intercept', value=1)
    # split data into train and test set
    print("splitting dataset into train and test sets...")
    X_train, X_test, y_train, y_test = tts(X, Y, test_size=0.2,
random_state=42)
    # train the model
    print("training started...")
    W = sgd(X_train.to_numpy(), y_train.to_numpy())
    print("training finished.")
    print("weights are: {}".format(W))
    # testing the model
    print("testing the model...")
    y_train_predicted = np.array([])
    for i in range(X_train.shape[0]):
```

```
        yp = np.sign(np.dot(X_train.to_numpy()[i], W))
        y_train_predicted = np.append(y_train_predicted, yp)
    y_test_predicted = np.array([])
    for i in range(X_test.shape[0]):
        yp = np.sign(np.dot(X_test.to_numpy()[i], W))
        y_test_predicted = np.append(y_test_predicted, yp)
    print("accuracy on test dataset:
{}".format(accuracy_score(y_test, y_test_predicted)))
    print("recall on test dataset:
{}".format(recall_score(y_test, y_test_predicted)))
    print("precision on test dataset:
{}".format(recall_score(y_test, y_test_predicted)))
# set hyper-parameters and call init
regularization_strength = 10000
learning_rate = 0.000001
init()
```

Appendix D - DNN

The speaker verification system studied in Chapter 4, uses a deep neural network (DNN) architecture as feature classifier. This appendix provides the python coding of the DNN architecture used for the feature classification of the speaker verification system is as follows:

```
import os, time, keras
        from keras import backend, metrics, callbacks, regularizers
        from keras.models import Sequential, model_from_json
        from keras.layers import Dropout, Flatten, BatchNormalization
        from keras.layers import Dense, Conv2D, MaxPooling2D
        from keras.utils import plot_model, multi_gpu_model
        import numpy as np
        from sklearn.metrics import classification_report
        def train(x_train, y_train, x_test, y_test, num_classes,
        epochs):
            # set validation data
            val_size = int(0.1 * x_train.shape[0])
            r = np.random.randint(0, x_train.shape[0], size=val_size)
            x_val = x_train[r, :, :]
            y_val = y_train[r]
            x_train = np.delete(x_train, r, axis=0)
            y_train = np.delete(y_train, r, axis=0)
            # set x & y shapes
            x_train = x_train.reshape(x_train.shape[0],
        x_train.shape[1], x_train.shape[2], 1)
            x_val = x_val.reshape(x_val.shape[0], x_val.shape[1],
        x_val.shape[2], 1)
            x_test = x_test.reshape(x_test.shape[0], x_test.shape[1],
        x_test.shape[2], 1)
            y_train = y_train.reshape(y_train.shape[0], 1)
            y_val = y_val.reshape(y_val.shape[0], 1)
            y_test = y_test.reshape(y_test.shape[0], 1)
            print('x_train shape:', x_train.shape)
            print('x_val shape:', x_val.shape)
            print('x_test shape:', x_test.shape)
            print('y_train shape:', y_train.shape)
            print('y_val shape:', y_val.shape)
            print('y_test shape:', y_test.shape)
            # convert class vectors to binary class matrices
            y_train = keras.utils.to_categorical(y_train, num_classes)
            y_val = keras.utils.to_categorical(y_val, num_classes)
            y_test = keras.utils.to_categorical(y_test, num_classes)
            # define the model
```

```
    regularizer=0.001
    dropout=0.25
    model = Sequential()
    model.add(Conv2D(16, kernel_size=(3, 3), strides=(2, 1),
activation='tanh',
kernel_regularizer=regularizers.l2(regularizer),
input_shape=(x_train.shape[1], x_train.shape[2], 1)))
    # model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 1)))
    model.add(Dropout(dropout))
    model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1),
activation='elu',
kernel_regularizer=regularizers.l2(regularizer)))
    # model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(dropout))
    model.add(Conv2D(64, kernel_size=(3, 3), strides=(1, 1),
activation='elu',
kernel_regularizer=regularizers.l2(regularizer)))
    # model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(dropout))
    model.add(Conv2D(64, kernel_size=(3, 3), strides=(1, 1),
activation='elu',
kernel_regularizer=regularizers.l2(regularizer)))
    # model.add(BatchNormalization())
    # model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(dropout))
    model.add(Flatten())
    model.add(Dense(64, activation='elu',
kernel_regularizer=regularizers.l2(regularizer)))
    model.add(Dropout(dropout))
    model.add(Dense(num_classes, activation='softmax'))
    # summarize model
    # plot_model(model, to_file='Models\\model_cnn2d.png')
    f = open('Models\\model_cnn2d.txt', 'w')
    for i in range(0, len(model.layers)):
        if i == 0:
            print(' ')
        print('{}. Layer {} with input / output shapes: {} /
{}'.format(i, model.layers[i].name, model.layers[i].input_shape,
model.layers[i].output_shape))
        f.write('{}. Layer {} with input / output shapes: {} /
{} \n'.format(i, model.layers[i].name,
model.layers[i].input_shape, model.layers[i].output_shape))
        if i == len(model.layers) - 1:
```

```
            print(' ')
    f.close()
    model.summary()
    # compile, fit evaluate
    try:    model = multi_gpu_model(model, gpus=2,
cpu_merge=False)
    except: model = model
    model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adam(), metrics = ['accuracy'])
    model.fit(x_train, y_train, batch_size=192, epochs=epochs,
verbose=2, validation_data=(x_val, y_val),
callbacks=[callbacks.EarlyStopping(monitor='val_acc',
patience=20, restore_best_weights=True)])
    time.sleep(5)
    score = model.evaluate(x_test, y_test, verbose=2)
    time.sleep(5)
    # save model
    '''model_json = model.to_json()
    with open("Models\\model_cnn2d.json", "w") as json_file:
        # remark
        json_file.write(model_json)
    model.save_weights("Models\\model_cnn2d.h5")'''
    # results
    # print('Test loss: {0:.3f} and accuracy:
{1:.3f}'.format(score[0], score[1]))
    print(' ')
    return score[1]
def predict(model, x_test):
    # load
    json_file = open('model_cnn2d.json', 'r')
    model = model_from_json(json_file.read())
    json_file.close()
    model.load_weights("model_cnn2d.h5")
    prediction = model.predict(x_test, batch_size=32, verbose=0)
    print(prediction)
```

Appendix E - ResNet

Deep residual learning (Resnet) has been used in the voice biometric based authentication model in Chapter 5. The python coding of the Resnet layer 34 architecture used in the model is provided in this appendix. Worth mentioning is that we have used TensorFlow as the python environment.

```
import tensorflow as tf
            from tensorflow import keras
            from keras.activations import relu
            from tensorflow.keras.layers import *
            from tensorflow.keras import Model
            from tensorflow.keras import layers as Layers
            class ResBlock(Model):
                def __init__(self, channels, stride=1):
                    super(ResBlock, self).__init__(name='ResBlock')
                    self.flag = (stride != 1)
                    self.conv1 = Conv2D(channels, 3, stride,
            padding='same')
                    self.bn1 = BatchNormalization()
                    self.conv2 = Conv2D(channels, 3, padding='same')
                    self.bn2 = BatchNormalization()
                    self.relu = ReLU()
                    if self.flag:
                        self.bn3 = BatchNormalization()
                        self.conv3 = Conv2D(channels, 1, stride)
                def call(self, x):
                    x1 = self.conv1(x)
                    x1 = self.bn1(x1)
                    x1 = self.relu(x1)
                    x1 = self.conv2(x1)
                    x1 = self.bn2(x1)
                    if self.flag:
                        x = self.conv3(x)
                        x = self.bn3(x)
                    x1 = Layers.add([x, x1])
                    x1 = self.relu(x1)
                    return x1
            class ResNet34(Model):
                def __init__(self):
                    super(ResNet34, self).__init__(name='ResNet34')
                    self.conv1 = Conv2D(64, 7, 2, padding='same')
                    self.bn = BatchNormalization()
                    self.relu = ReLU()
                    self.mp1 = MaxPooling2D(3, 2)
```

```
        self.conv2_1 = ResBlock(64)
        self.conv2_2 = ResBlock(64)
        self.conv2_3 = ResBlock(64)
        self.conv3_1 = ResBlock(128, 2)
        self.conv3_2 = ResBlock(128)
        self.conv3_3 = ResBlock(128)
        self.conv3_4 = ResBlock(128)
        self.conv4_1 = ResBlock(256, 2)
        self.conv4_2 = ResBlock(256)
        self.conv4_3 = ResBlock(256)
        self.conv4_4 = ResBlock(256)
        self.conv4_5 = ResBlock(256)
        self.conv4_6 = ResBlock(256)
        self.conv5_1 = ResBlock(512, 2)
        self.conv5_2 = ResBlock(512)
        self.conv5_3 = ResBlock(512)
        self.pool = GlobalAveragePooling2D()
        self.fc1 = Dense(512, activation='relu')
        self.dp1 = Dropout(0.5)
        self.fc2 = Dense(512, activation='relu')
        self.dp2 = Dropout(0.5)
        self.fc3 = Dense(64)
    def call(self, x):
        x = self.conv1(x)
        x = self.bn(x)
        x = self.relu(x)
        x = self.mp1(x)
        x = self.conv2_1(x)
        x = self.conv2_2(x)
        x = self.conv2_3(x)
        x = self.conv3_1(x)
        x = self.conv3_2(x)
        x = self.conv3_3(x)
        x = self.conv3_4(x)
        x = self.conv4_1(x)
        x = self.conv4_2(x)
        x = self.conv4_3(x)
        x = self.conv4_4(x)
        x = self.conv4_5(x)
        x = self.conv4_6(x)
        x = self.conv5_1(x)
        x = self.conv5_2(x)
        x = self.conv5_3(x)
        x = self.pool(x)
        x = self.fc1(x)
        x = self.dp1(x)
```

```
        x = self.fc2(x)
        x = self.dp2(x)
        x = self.fc3(x)
        return x
model = ResNet34()
model.build(input_shape=(1, 480, 480, 3))
model.summary()
```