Voice biometric system security: Design and analysis of countermeasures for replay attacks

Bhusan Chettri

PhD thesis

School of Electronic Engineering and Computer Science
Queen Mary University of London

2020

Author's declaration

I, Bhusan Chettri, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material are also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Rights, or contain any confidential material.

I accept that the university has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature: Bhusan Chettri
Date: 19/05/2020

Details of collaboration and publications: see Section 1.5.

Abstract

Voice biometric systems use *automatic speaker verification* (ASV) technology for user authentication. Even if it is among the most convenient means of biometric authentication, the robustness and security of ASV in the face of *spoofing attacks* (or *presentation attacks*) is of growing concern and is now well acknowledged by the research community. A spoofing attack involves illegitimate access to personal data of a targeted user. *Replay* is among the simplest attacks to mount — yet difficult to detect reliably and is the focus of this thesis.

This research focuses on the analysis and design of existing and novel countermeasures for replay attack detection in ASV, organised in two major parts. The first part of the thesis investigates existing methods for spoofing detection from several perspectives. I first study the generalisability of hand-crafted features for replay detection that show promising results on synthetic speech detection. I find, however, that it is difficult to achieve similar levels of performance due to the acoustically different problem under investigation. In addition, I show how class-dependent cues in a benchmark dataset (ASVspoof 2017) can lead to the manipulation of class predictions. I then analyse the performance of several countermeasure models under varied replay attack conditions. I find that it is difficult to account for the effects of various factors in a replay attack: acoustic environment, playback device and recording device, and their interactions. Subsequently, I developed and studied a convolutional neural network (CNN) model that demonstrates comparable performance to the one that ranked first in the ASVspoof 2017 challenge. Here, the experiment analyses what the CNN has learned for replay detection using a method from interpretable machine learning. The findings suggest that the model highly attends at the first few milliseconds of test recordings in order to make predictions. Then, I perform an in-depth analysis of a benchmark dataset (ASVspoof 2017) for spoofing detection and demonstrate that any machine learning countermeasure model can still exploit the artefacts I identified in this dataset.

The second part of the thesis studies the design of countermeasures for ASV, focusing on model robustness and avoiding dataset biases. First, I proposed an ensemble model combining shallow and deep machine learning methods for spoofing detection, and then demonstrate its effectiveness on the latest benchmark datasets (ASVspoof 2019). Next, I proposed the use of speech endpoint detection for reliable and robust model predictions on the ASVspoof 2017 dataset. For this, I created a publicly available collection of hand-annotations of speech

endpoints for the same dataset, and new benchmark results for both frame-based and utterance-based countermeasures are also developed.

I then proposed spectral subband modelling using CNNs for replay detection. My results indicate that models that learn subband-specific information substantially outperform models trained on complete spectrograms. Finally, I proposed to use variational autoencoders — deep unsupervised generative models — as an alternative backend for spoofing detection and demonstrate encouraging results when compared with the traditional Gaussian mixture models.

Acknowledgements

As the journey of my PhD research has come to an end, I look back and realize that this wouldn't have been possible without the blessing from the lord almighty and the tremendous support and guidance from the exceptional supervisors I was fortunate to have met during this journey.

First and foremost, I would like to thank my supervisor Dr. Bob L. Sturm for giving me the freedom to explore the research work on voice-biometrics security although we had agreed to work on a different topic in the start. His sound advice and motivation has helped me to improve my research skills and grow as a researcher.

I would also like to thank my supervisor Dr. Emmanouil Benetos for kindly agreeing to supervise me from the second year of my PhD and providing me with all the best guidance I needed to finish this PhD research. His prompt way of responding to my questions, detailed feedback, and supervision has helped me to accomplish the goals of my thesis. Without his support and guidance, this journey would have been a difficult one.

I would also like to thank Prof. Ioannis Patras and Dr. Dan Stowell for their detailed feedback and useful advice that helped shape my research work. I want to thank all the members of C4DM for providing a wonderful workplace with a willingness to always help which really helped me to overcome stressful moments, and stay focused towards achieving my research goals on time.

My sincere thanks goes to Dr. Tomi Kinnunen who introduced me the research topic on the security of voice-biometrics. I am extremely grateful to him for his mentorship and support throughout my PhD and for the opportunity he gave me to work on the NOTCH project at the University of Eastern Finland. Special thanks to all the members of Computational speech group and Juha Hakkarainen for their support during my stay in Finland.

I want to thank my wife, Namrata Bora for all her tremendous support and understanding throughout this journey, without which none of this would have been possible. I would also like to to thank my family back in India, my father Tilak Chettri, mother Krishna Maya Chettri, my in-laws Kiran Chandra Bora

and Rekha Bora, my sisters (Roshni and Seema), Pinki Bora, Chandan Bora and our little Tia for their motivation through love and prayers for good health during my PhD study.

I am also very grateful to George and Christine, they have played a major role in helping me make a big decision of considering QMUL for my PhD over other offers I was holding.

Finally, I take this opportunity to thank Chinedu Iheagwam for his help on proof-reading my thesis.

This PhD research work was funded by a QMUL Principal's research studentship.

Contents

List of Figures

List of Tables

List of abbreviations

ASV	Automatic Speaker Verification
ASR	Automatic Speech Recognition
CNN	Convolutional Neural Network
CQT	Constant Q Transform
CQCC	Constant Q Cepstral Coefficient
CM	Countermeasure
CVAE	Conditional Variational Autoencoder
DNN	Deep Neural Network
DFT	Discrete Fourier Transform
ELBO	Evidence Lower Bound
EER	Equal Error Rate
EM	Expectation Maximization
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
IMFCC	Inverted Mel-Frequency Cepstral Coefficient
KL	Kullback Leibler
LA	Logical Access
LDA	Linear Discriminant Analysis
LFCC	Linear Frequency Cepstral Coefficient
MFCC	Mel-Frequency Cepstral Coefficient
ML	Machine Learning
MFM	Max Feature Map
PCA	Principal Components Analysis
PLDA	Probabilistic Linear Discriminant Analysis
PA	Physical Access
PDF	Probability Density Function
RELU	Rectified Linear Units
RFCC	Rectangular Frequency Cepstral Coefficient
SCMC	Subband Centroid Magnitude Coefficient
SVM	Support Vector Machine
TTS	Text to Speech Synthesis
t-DCF	Tandem Detection Cost Function
UBM	Universal Background Model
VC	Voice Conversion
VAE	Variational Autoencoder

Chapter 1

Introduction

The main topic of this thesis is analysis and design of spoofing countermeasures for secure voice biometrics. This chapter first explains the motivations and aims of this work in Sections 1.1 and 1.2. Then the structure of the thesis is described in Section 1.3. Finally, Section 1.4 summarises the publications associated with this thesis.

1.1 Motivation

Voice biometric systems use *automatic speaker verification* (ASV) [Reynolds, 1995] technology for user authentication. The main goal of an ASV system is to verify the identity of a claimed person using their voice characteristics. Even if it is among the most convenient means of biometric authentication, the robustness and security of ASV in the face of *spoofing attacks* (or *presentation attacks*) is of growing concern, and is now well acknowledged by the community [Sahidullah et al., 2019]. A spoofing attack involves illegitimate access to personal data of a targeted user. The vulnerability of ASV systems against spoofing attacks is an important problem to solve because it poses a serious threat to the security of such systems. When successful, a spoofing attack can grant unauthorized access of private and sensitive data. Spoofing attack methods include text-to-speech (TTS) [Masuko et al., 1999], voice conversion (VC) [Pellom and Hansen, 1999] techniques, impersonation [Lau et al., 2004] and playing back speech recordings [Wu et al., 2014a]. Section 2.2 provides further background on spoofing attacks.

High-stakes ASV applications, therefore, demand trustworthy fail-safe mechanisms (countermeasures) against such attacks. In this thesis, a *countermeasure* (CM) is defined as a binary classifier that aims at discriminating *bonafide* (human speech) utterances from spoofing attacks. To allow maximum re-usability across different applications, the ideal CM should generalise across environ-

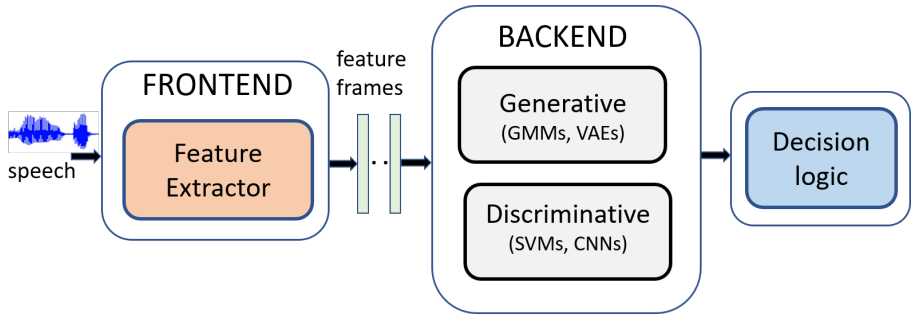


Figure 1.1: Countermeasure for spoofing attack detection.

ments, speakers, languages, channels, and attacks. In practice, this is not the case; CMs are prone to overfitting. This could be due to variations within the spoof class (*e.g.* speech synthesizers or attack conditions not present in the training set), within the bonafide class (*e.g.* due to content and speaker), or extrinsic nuisance factors (*e.g.* background noise).

Like any traditional machine learning classifier, a spoofing countermeasure (Fig. 1.1) typically consists of a frontend module, a backend module and a decision logic (for the final classification based on a decision threshold). The key function of the frontend is to transform the raw acoustic waveform to a sequence of *short-term feature vectors*. These short-term feature vectors are then used to derive either intermediate recording-level features (such as *i-vectors* [Khoury et al., 2014] or *x-vectors* [Williams and Rownicka, 2019]) or statistical models, such as *Gaussian mixture models* (GMMs) [Patel and Patil, 2015] to be used for bonafide or spoof class modeling. In contrast to these approaches that require a certain level of handcrafting especially in the frontend, modern *deep-learning* based countermeasures are often trained using either raw-audio waveforms [Dinkel et al., 2017] or an intermediate high-dimensional time-frequency representation — often the power spectrogram [Zhang et al., 2017]. In these approaches, the notions of frontend and backend are less clearly distinguished. A summary of deep learning-based methods for spoofing detection is provided in Subsection 2.4.2 and Table 2.2.

Among four different approaches of spoofing ASV systems highlighted early in this section, this thesis focusses on replay attacks. The motivations are twofold. First, replay attack is the simplest form of attack to implement that does not require any specific expertise either from speech technology or machine learning. Second, it is equally difficult to detect reliably as this form of attack involves propagating the recorded speech of a target user to fool an ASV system. Two publicly available replay datasets, ASVspoof 2017 and ASVspoof 2019 PA (described in Chapter 3), are used to study replay attacks in this thesis.

1.2 Aim

The aim of this thesis is to analyse and design existing and novel methods for replay spoofing detection for secure voice biometrics. To this end, this thesis focusses on answering the following research questions:

- Can hand-crafted features used in speech processing be used for replay spoofing detection?
- Are dataset artefacts biasing model predictions?
- Can data-driven ML models be effective in discriminative spoofing attacks? Is the current state-of-the-art model for replay spoofing detection reproducible?
- Can countermeasures designed using speech endpoint detection (discarding everything before and after the actual speech utterance) ensure robustness and good detection performance?
- What makes replay spoofing detection hard? How do different factors such as acoustic environment, recording device and playback device affect the performance of replay countermeasures?
- Can we use methods from interpretable machine learning to understand predictions of countermeasure models?
- How do data selection for model training and validation impact model generalisation? Would combining information from multiple models improve performance?
- Can information exploited in specific subbands be exploited to discriminate between bonafide and spoof recordings?
- Can Variational Autoencoders (VAEs), a deep generative model, be used as a backend classifier for spoofing detection? Can VAEs be used to derive new feature representations for replay spoofing detection?

The research involved in addressing the above listed questions is presented in two major chapters: Chapter 4 and Chapter 5.

1.3 Thesis structure

- **Chapter 2** provides the necessary background on spoofing attacks and methods used for spoofing detection in the literature. It starts with a brief overview of classical and deep learning-based ASV systems. A detailed survey of spoofing attacks and different countermeasures based on signal processing and machine learning methods are presented. The chapter also summarises the related literature on subband modelling for spoofing detection. Finally, this chapter provides a detailed background on variational autoencoders along with the motivation towards using them for spoofing detection.

- **Chapter 3** presents details of the corpus and the evaluation metrics used in this thesis. Firstly, it provides a detailed summary of the ASVspoof 2017 and ASVspoof 2019 datasets used for the analysis and design of spoofing countermeasures. A brief overview of other publicly available spoofing datasets is also provided for general awareness. Finally, the chapter explains the two metrics that are adopted in this thesis to evaluate the performance of a countermeasure.

- **Chapter 4** serves as the basis towards understanding the replay spoofing problem by investigating existing methods from the literature. Firstly, signal processing methods used in TTS and VC spoofing detection are studied. The chapter then summarises issues towards reproducing a state-of-the-art CNN model, proposes an adapted version of this model, and analyses it to discover cues on which it focusses for classification. The chapter also analyses the effect of different factors involved in a replay attack and their interactions. Then, an in-depth analysis on a benchmark dataset highlighting dataset artefacts and their influence on model decisions is provided.

- **Chapter 5** presents novel methods proposed for replay spoofing detection, with a focus on model robustness and avoiding biases in the datasets. Ensemble models and dataset partitions are first proposed for improved generalisation. Speech endpoint detection for reliable performance estimates is proposed. Using this, a frame-level deep countermeasure model is proposed showcasing its robustness against cues in the dataset. Further, a joint subband framework that exploits information across different frequency bands of a spectrogram is proposed. Finally, this chapter proposes VAEs as an alternative backend classifier and as a feature extractor.

- **Chapter 6** concludes this thesis. It provides a summary of research find-

ings related to the analysis and design of replay attack countermeasures in this thesis. A general discussion on future research directions towards improved countermeasure design and challenges involved are also provided.

1.4 Contributions

The key contributions of the research carried out in this thesis are summarised below:

Chapter 3

- A qualitative analysis of the ASVspoof v2.0 dataset.

Chapter 4

- A study on the effectiveness of existing signal processing methods for replay spoofing detection.
- Demonstration of the challenges in replicating a state-of-the-art deep model for replay attack detection, a study on alternative deep models, and an investigation regarding network hyper-parameters.
- An analysis of the developed model using a method from interpretable machine learning to understand what it has learned to detect spoof recordings.
- An analysis of countermeasure model performance under varied attack conditions, highlighting the difficulty in understanding the influence of different conditions and their interaction in a replay attack.
- Discovery of dataset-related artefacts in a benchmark replay spoofing dataset (ASVspoof 2017 v2.0) and a detailed analysis demonstrating awareness on how they influence machine learning models.

Chapter 5

- An ensemble model combining deep and shallow models for effective spoofing detection.
- Dataset partitions for model training and validation for better generalisation.
- Discovery of dataset-related cues in the latest benchmark spoofing dataset (ASVspoof 2019 PA) that models exploit in decision making.

- Countermeasure design using speech endpoint detection for robust performance estimates, new benchmark results, and a new frame-level robust deep countermeasure model.
- Manual and automatic speech endpoint annotations for the ASVspoof 2017 v2.0 dataset.
- A joint subband modelling framework using CNNs for effective spoofing detection.
- Experimental evaluation on how different subbands contribute in spoofing detection, and their cross-dataset performance evaluation on an unseen testset.
- A deep generative model as an alternative backend to traditional shallow generative models.
- New features for replay spoofing detection using deep generative models.

1.5 Associated publications

This thesis covers work on replay spoofing detection which was carried out by the author between September 2016 and December 2020 at Queen Mary University of London. The work on variational autoencoders for spoofing detection (detailed in Section 5.5) was performed during a six-month (May - October 2019) research visit to the Computational Speech group, University of Eastern Finland, Finland. The majority of the work presented in this thesis has been presented in international peer-reviewed conferences and journals:

Journal papers

[1] B. Chettri, T. Kinnunen and E. Benetos, "Deep Generative Variational Autoencoding for Replay Spoof Detection in Automatic Speaker Verification", *Computer Speech & Language*, vol. 63, September 2020.

[2] B. Chettri, E. Benetos and B. L. Sturm, "Dataset Artefacts in Anti-Spoofing Systems: a Case Study on the ASVspoof 2017 benchmark", submitted to *IEEE/ACM Transactions on Audio Speech and Language Processing*.

Conference papers

[3] B. Chettri and B. L. Sturm, "A Deeper Look at Gaussian Mixture Model Based Anti-Spoofing Systems", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5159-5163, April 2018.

[4] B. Chettri, B. L. Sturm and E. Benetos, "Analysing Replay Spoofing Countermeasure Performance Under Varied Conditions", in *IEEE* 28^{th} *International Workshop on Machine Learning for Signal Processing (MLSP)*, September 2018.

[5] B. Chettri, S. Mishra, B. L. Sturm and E. Benetos, "Analysing the Predictions of a CNN-Based Replay Spoofing Detection System", in *IEEE Spoken Language Technology Workshop (SLT)*, pp. 92–97, December 2018.

[6] B. Chettri, D. Stoller, V. Morfi, M. A. M. Ramírez, E. Benetos and B. L. Sturm, "Ensemble Models for Spoofing Detection in Automatic Speaker Verification", in *Proc. Interspeech*, pp. 1018–1022, September 2019.

[7] B. Chettri, T. Kinnunen and E. Benetos, "Subband Modeling for Spoofing Detection in Automatic Speaker Verification", to appear in *Odyssey 2020: The Speaker and Language Recognition Workshop*, November 2020.

Other publications

[8] B. Chettri, S. Mishra, B. L. Sturm and E. Benetos, "A Study on Convolutional Neural Network Based End-To-End Replay Anti-Spoofing", *arXiv:1805.09164* preprint, May 2018.

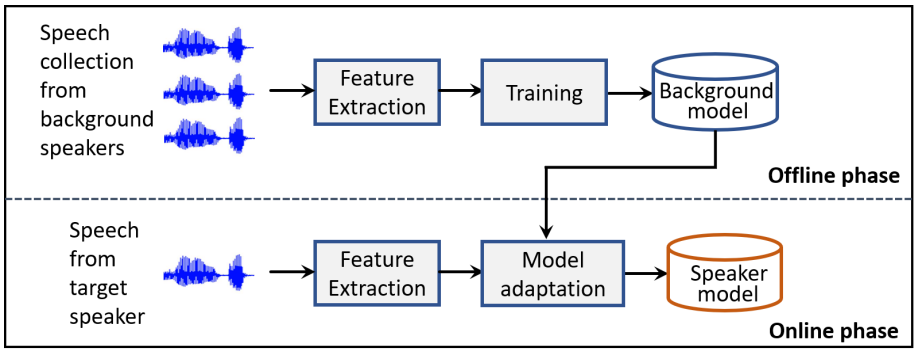
Here we provide a summary of contributions from authors listed in the above publications. For [5], S. Mishra contributed in writing the technical description of the SLIME algorithm and also contributed to the discussions on the intervention experiments. For [6], each of the co-authors D. Stoller, V. Morfi, M. A. M. Ramírez contributed in training one deep model (and writing the technical description of their respective models) that were used in building the ensemble model. Furthermore, V. Morfi contributed in preparing the proposed partition for the PA dataset, and D. Stoller helped in reviewing the paper draft. For [8], S. Mishra helped in reviewing the paper and contributed to the discussion on replicating a state-of-the-art CNN model.

In all of the above publications, the thesis author is the main contributor responsible for the design and implementation of experiments, analysis of results and writing the papers (as the lead author). Co-authors Dr. T. Kinnunen, Dr. B. L. Sturm and Dr. E. Benetos have contributed in a supervisory role towards discussion of research ideas, analysing the results of research experiments, and helping with reviewing the drafts of the papers. Furthermore, Dr. T. Kinnunen also contributed in writing the theory section of the journal article in [1].

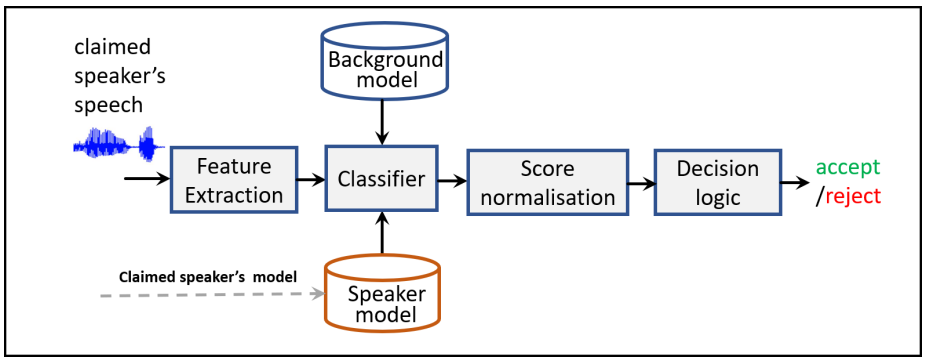
Chapter 2

Background

This chapter describes state-of-the-art methods on spoofing countermeasures for voice biometric systems. Voice biometric systems use automatic speaker verification (ASV) technologies. Therefore, this chapter first provides background information on classical and state-of-the-art ASV systems (Section 2.1). Then Section 2.2 provides a description of various spoofing attacks that are used to bypass an ASV system. A summary of the community driven Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVspoof) is provided in Section 2.3, which focusses on promoting anti-spoofing research while releasing standard evaluation protocols and publicly available databases. Following this, the next Section 2.4 provides a detailed literature on countermeasures for replay spoofing attacks that were studied before the ASVspoof series began along with published works on the ASVspoof datasets for replay attack. For better readability, this literature is organized into two categories: traditional methods (Subsection 2.4.1) and deep learning methods (Subsection 2.4.2). Then the next Section 2.5 provides a background on various hand-crafted features that have been used in this thesis. After this, a detailed background on backend classifiers that have been investigated towards the analysis and design of replay spoofing countermeasures is provided in Sections 2.6 and 2.7. Section 2.6 provides a background on discriminative models such as support vector machines and convolutional neural networks. Section 2.7 on the other hand provides background on generative models such as Gaussian mixture models, i-vectors and Variational Autoencoders. The next Section 2.8 summarises the related works on subband modelling applied to audio and speech applications including spoofing detection. Then Section 2.9 describes work on trustworthy machine learning and its significance on countermeasures for spoofing detection. First, information on how artefacts and confounders in a dataset can affect model predictions is provided in Subsection 2.9.1. Then Subsection 2.9.2 explains a method from



(a) **Speaker enrollment phase**. The goal here is to build speaker specific models by adapting a background model which is trained on a large speech database.



(b) **Speaker verification phase**. For a given speech utterance the system obtains a verification score and makes a decision whether to accept or reject the claimed identity.

Figure 2.1: Components of a typical speaker verification system. Figure adapted from [Kinnunen and Li, 2010b].

interpretable machine learning called SLIME that is used in this thesis to understand the predictions of a CNN-based countermeasure in Section 4.5. Finally, Section 2.10 concludes this chapter.

2.1 Automatic speaker verification (ASV)

Automatic speaker verification (ASV) [Reynolds, 1995] systems aim at verifying the identity of a claimed person using their voice characteristics. They are commonly used in user authentication and surveillance applications. Fig. 2.1 illustrates components of a typical speaker verification system which comprises speaker enrollment (top) and speaker verification (bottom). The role of a feature extraction module is to transform the raw speech signal into some representation (features) that retains speaker specific attributes useful to the downstream components in building speaker models. The enrollment phase comprises offline and

online modes of building models. During the offline mode, background models are trained on features computed from a large speech collection representing a diverse population of speakers. The online phase comprises building a target speaker model using features computed from target speaker's speech. Usually, training the target speaker model from scratch is avoided because learning reliable model parameters requires a sufficiently large amount of speech data, which is usually not available for every individual speaker. To overcome this, the parameters of a pretrained background model representing the speaker population are adapted using the speaker data yielding a reliable speaker model estimate. During the speaker verification phase, for a given test speech utterance, a claimed speaker's model and the background model (representing the world of all other possible speakers) is used to derive a confidence score. The decision logic module then makes a binary decision: it either accepts the claimed identity as a genuine speaker or rejects it as an impostor based on some decision threshold.

ASV systems are broadly grouped into two categories: text dependent and text independent systems. In text dependent systems [Larcher et al., 2014], the same speech utterance used during speaker enrollment (or registration) is used during verification (or testing). This is in contrast with text independent systems [Kinnunen and Li, 2010b], where as the name suggests, speakers are free to speak any arbitrary phrase during training and testing. Text independent systems are in general difficult in contrast to text dependent systems due to the variability in spoken utterances during the testing phase. We now provide a high-level discussion on the traditional and current deep learning approaches to speaker verification.

Traditional methods. By traditional methods we refer to approaches driven by a Gaussian mixture model - universal background model (GMM-UBM) [Kinnunen and Li, 2010b] that were adopted in the ASV literature until deep learning techniques became popular in the field. Mel-frequency cepstral coefficients (MFCCs) [Davis and Mermelstein, 1980] were popular frame-level feature representations used in speaker verification. Using short-term MFCC feature vectors, utterance level features such as i-vectors are often derived which have shown state-of-the-art performance in speaker verification [Kanagasundaram, 2014]. Subsection 2.5 describes further details on the steps involved in computing MFCCs. The background models such as the Universal background model (UBM) and total variability (T) matrix are learned in an offline phase using a large collection of speech data. The UBM and T matrix are used in computing i-vector representations. Subsection 2.7.2 provides further details on i-vectors. The training process involves learning model (target or background) parameters from training data. As for modelling techniques, vec-

tor quantization (VQ) [Burton, 1987] was one of the earliest approaches used to represent a speaker, after which Gaussian mixture models (GMMs), an extension to VQ methods, and Support vector machines (Subsection 2.6.1) became popular methods for speaker modelling. Please see [Kinnunen and Li, 2010b] for a detailed overview on traditional methods. The traditional approach also includes training an i-vector extractor (GMM-UBM, T-matrix) on MFCCs and using a probabilistic linear discriminant analysis (PLDA) [Ioffe, 2006] backend for scoring.

Deep learning methods. We now provide a brief summary on deep learning approaches adopted in ASV. Features are often learned in a data-driven manner directly from the raw speech signal [Jung et al., 2018] or from some intermediate speech representations such as filter bank energies [Heigold et al., 2016]. Handcrafted features, for example MFCCs, are often used as input to train deep neural network (DNN) based ASV systems [Kenny et al., 2014]. Features learned from DNNs are often used to build traditional ASV systems [Yaman et al., 2012, Snyder et al., 2017]. Snyder et al. [2017] uses the output from the penultimate layer of a pre-trained DNN as features to train a traditional i-vector PLDA setup (replacing i-vectors with DNN features). Yaman et al. [2012] extracts bottleneck features (output from a hidden layer with a relatively small number of units) from a DNN to train a GMM-UBM system which uses the log-likelihood ratio as scoring.

Utterance-level discriminative features, so called embeddings extracted from pre-trained DNNs have become popular recently, demonstrating good results. Examples of such embeddings include the d-vector [Variani et al., 2014] and x-vector [Snyder et al., 2018], and have been used in many works, for example [Snyder et al., 2017, Garcia-Romero et al., 2019]. End-to-end modelling approaches have also been extensively studied in speaker verification showing promising results [Jung et al., 2018, Muckenhirn et al., 2018, Ravanelli and Bengio, 2018]. In this setting, both feature learning and model training are jointly optimised from the raw speech input. A wide range of neural architectures have been studied for speaker verification. This includes feed forward neural networks, commonly referred as deep neural networks (DNNs) [Variani et al., 2014, Sztahó et al., 2019], convolutional neural networks (CNNs) [Muckenhirn et al., 2018], recurrent neural networks [Tang et al., 2019], and attention models [Rezaur rahman Chowdhury et al., 2018].

Training background models in deep learning approaches can be thought of as a pretrainng phase where network parameters are trained on a large dataset. Speaker models are then derived by adapting the pretrained model parameters using speaker specific data, much like the same way a traditional GMM-UBM system operates. For example, [Muckenhirn et al., 2018] first trains a DNN to

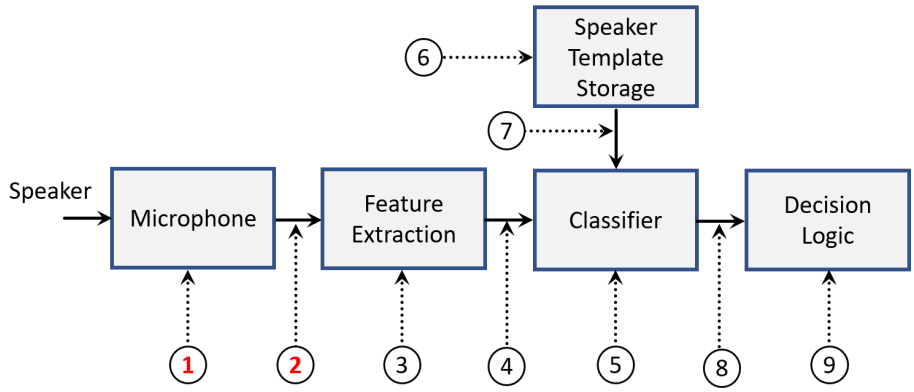


Figure 2.2: Possible locations [ISO/IEC, 2016] to attack an ASV system. 1: microphone point, 2: transmission point, 3: override feature extractor, 4: modify features, 5: override classifier, 6: modify speaker database, 7: modify biometric reference, 8: modify score and 9: override decision.

perform speaker identification discriminatively. In the second step, they use the pretrained model to derive a speaker specific model. They repeat this process for every speaker using speaker specific data.

As the main focus of this thesis is not on ASV, we have provided only a high-level introductory overview on ASV to familiarise readers with the basics of voice biometrics, with pointers to relevant papers for additional details. Please refer to [Kinnunen and Li, 2010b] and [Sztahó et al., 2019] for further background on traditional and deep learning approaches for ASV. Furthermore, the reader is referred to [Kanagasundaram, 2014] for a detailed background on i-vector based ASV systems.

2.2 Spoofing attacks in ASV

A spoofing attack (or *presentation attack*) involves illegitimate access to the personal data of a targeted user. These attacks are performed on a biometric system to provoke an increase in its false acceptance rate. The security threats imposed by such attacks are now well acknowledged within the speech community [Sahidullah et al., 2019]. As identified in the ISO/IEC 30107-1 standard [ISO/IEC, 2016], a biometric system could be potentially attacked from nine different points. Fig. 2.2 summarises this. The first two attack points are of specific interest as they are particularly vulnerable in terms of enabling an adversary to inject spoofed biometric data. These two points are commonly referred as *physical access* (PA) and *logical access* (LA) attacks. As illustrated in the figure, PA attacks involve presentation attack at the sensor (microphone

in case of ASV) level and LA attacks involve modifying biometric samples to bypass the sensor. As highlighted in Section 1.1, TTS [Masuko et al., 1999] and VC [Pellom and Hansen, 1999] techniques are used to produce artificial speech to bypass an ASV system. These two methods are examples of LA attacks. On the other hand, mimicry [Lau et al., 2004] and playing back speech recordings (replay) [Wu et al., 2014b]) are examples of PA attacks.

2.2.1 Mimicry

This form of attack involves an attacker attempting to modify their voice characteristics to sound like a target speaker. In other words, an attacker aims to transform their lexical and prosodic properties to be able to sound as close as possible to the target speaker [Lau et al., 2004, Sahidullah et al., 2019]. Therefore, this form of attack can be highly effective when the attacker's voice is similar to the target speaker, as less effort would be required to adjust the voice of an attacker in contrast to situations where the voice of the attacker is less similar to the target speaker [Lau et al., 2005]. In other words, the success of mimicry attacks often depends on the degree or quality of the impersonated voice, suggesting that professional impersonators may be better at mimicking a target speaker's voice than inexperienced impersonators [Mariéthoz and Bengio, 2005]. Furthermore, successful attackers were found to be able to transform their F0 (fundamental frequency) and sometimes the formants close to the target speaker [Perrot et al., 2005, Zetterholm, 2007].

While there have been some studies assessing the threat of mimicry to ASV systems [Hautamäki et al., 2015], the main challenge however is the unavailability of large corpora of impersonated speech. Due to this, research on mimicry attacks is still behind in contrast to TTS, VC and replay attacks for which large public corpora and standard evaluation protocols (See Chapter 3) are available, which has promoted growth in anti-spoofing research for these attacks [Wu et al., 2015c, Kinnunen et al., 2017a, Todisco et al., 2019.]. Therefore, the true potential threat of mimicry on ASV is still not very clear [Sahidullah et al., 2019].

2.2.2 Speech synthesis

Speech synthesis or text-to-speech (TTS), is a method to generate speech from a given text input that sounds as natural and intelligible as possible. It has a wide range of applications including spoken dialogue systems, speech-to-speech translation, assisting people with vocal disorders, and automatic e-book reading, to name a few [Taylor, 2009]. Text analysis and speech waveform generation are the two main components of a typical TTS system. The text analysis com-

ponent analyses the input text and produces sequence of phonemes defining the linguistic specification of the text. Using these phonemes, the speech waveform generation module produces the speech waveform [Hunt and Black, 1996, Zen et al., 2009]. However, in end-to-end deep learning frameworks, speech waveforms are directly generated from the input text [Gibiansky et al., 2017]. An early work investigating the impact of synthetic speech on the performance of an ASV system is [Masuko et al., 1999]. The authors used a hidden Markov model (HMM) based synthetic speech to fool an HMM-based ASV system, demonstrating an increased false acceptance rate. Please refer to [Wu et al., 2015b] and [Sahidullah et al., 2019] for additional background on TTS spoofing and countermeasures.

2.2.3 Voice conversion

Voice conversion aims at converting the voice of a speaker to that of another. In the context of ASV spoofing, the source voice corresponds to an attacker which is converted to that of a target speaker to fool an ASV system. Typical VC systems operate directly on speech signals of the source and target speaker using a parallel corpus of the two speakers (speaking the same utterances) on which a transformation function is learned to convert the attacker acoustic parameters to that of a target speaker [Mohammadi and Kain, 2017]. Applications of VC technologies include producing natural sounding voices for people with speech disabilities and voice dubbing in entertainment industries to name a few. Some of the early works demonstrating the impact of voice converted speech to fool an ASV system include that of [Pellom and Hansen, 1999, Matrouf et al., 2006]. Furthermore, Kinnunen et al. [2012] demonstrated the impact of converted speech across a wide range of ASV systems. They used a joint density Gaussian mixture model (JDGMM) based VC system to fool ASV systems showing increased error rates. Please refer to [Desai et al., 2009, Fang et al., 2018a, Wu et al., 2015b, Sahidullah et al., 2019] for additional background on VC.

2.2.4 Replay attacks

A replay spoofing attack involves playing back recorded speech samples of a target speaker (enrolled speaker) to bypass an ASV system. This type of attack requires physical transmission of spoofed speech through the system microphone. This is shown as point 1 in Fig. 2.2. Replay is the simplest form of a spoofing attack that can be implemented using smartphones, and does not require specific expertise either in speech processing or machine learning techniques. Fig. 2.3 illustrates the difference between a bonafide/genuine speech

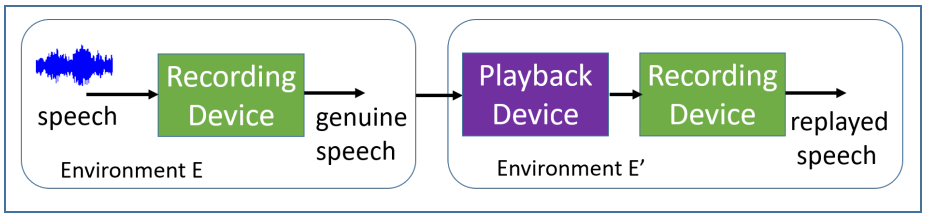


Figure 2.3: Difference between a genuine (bonafide) and a replayed speech.

and a replayed speech signal. Here, bonafide/genuine speech corresponds to speech spoken by a target speaker during enrollment (or the verification phase) and is acquired by an ASV system's microphone. On the other hand, a replayed speech denotes the speech signal that is obtained by playing back a pre-recorded bonafide speech which is then acquired by the system's microphone. The acoustic environment for the acquisition of bonafide speech, and the replayed speech can be the same — situations where an attacker manages to launch the attack from the same physical space. But, in practice the acoustic space is usually different (eg. a different closed room/office with no background noise) as an attacker would not want to risk getting caught while launching such attacks. Therefore, factors of interest in detecting replay attacks are changes/noise induced in bonafide speech from the loudspeaker of playback device, recording device and the acoustic environment where the replay attack is simulated.

Section 2.4 will provide an in-depth description of countermeasures for replay attacks proposed in the literature before and after the release of standard spoofing datasets (Section 3) released as part of the ASVspoof open spoofing evaluations. Before that, the next Section 2.3 will provide a summary of the ASVspoof challenge series that focusses on promoting anti-spoofing research for secure voice biometrics.

2.3 ASVspoof challenge

ASVspoof[1], the automatic speaker verification spoofing and countermeasures challenge, is an ASV community driven effort promoting research in developing anti-spoofing algorithms for secure voice biometrics. As summarised in [Wu et al., 2015b], a number of research studies had confirmed the vulnerability of voice biometrics to spoofing attacks, before the ASVspoof series began in 2015. However, these studies were mostly performed on small in-house datasets comprising limited speakers and spoofing attack conditions. Therefore, research results were hard to reproduce and understanding the true generalisability of the

[1]https://www.asvspoof.org/

reported anti-spoofing solutions in unseen attack conditions was difficult. The main motivation of the ASVspoof series was to overcome these issues by organizing open spoofing challenge evaluations, promoting awareness of the problem, making publicly available spoofing corpora comprising sufficiently varying attack conditions with standard evaluation protocols, and further ensuring transparent research leading to reproducible results.

The first ASVspoof challenge held in 2015 focused on the detection of artificial speech generated using either speech synthesis (TTS) or voice conversion (VC) algorithms in a text-independent setting. Clean speech recorded using high quality microphones was used as bonafide speech and seven VC and three TTS algorithms were used to produce spoofed speech [Wu et al., 2015c]. The second edition of the ASVspoof challenge held in 2017 focussed on text-dependent replay spoofing attack detection [Kinnunen et al., 2017a]. Section 3.2 describes details of the dataset used in the ASVspoof 2017 evaluation. The recent edition held in 2019, ASVspoof 2019, combined both TTS, VC and replay attacks together, using advanced state-of-the art spoofing algorithms and methods to generate spoofed speech samples [Todisco et al., 2019., Wang et al., 2019b]. Section 3.3 provides further details on the datasets used in the 2019 challenge.

One key observation that is worth noting from the three ASVspoof challenges is the paradigm shift in the use of modelling approaches for spoofing detection. Gaussian mixture models (GMMs), which is a generative model, were popular during the first ASVspoof challenge in 2015 as evident from the winning system of this challenge which is a GMM-based system [Patel and Patil, 2015]. However, the 2017 and 2019 spoofing challenges were mostly dominated by data-driven discriminatively trained deep models [Lavrentyeva et al., 2017, 2019]. Section 2.4 provides further details. The main task, however, in all the three editions of the ASVspoof challenge was to build a standalone countermeasure model (anti-spoofing algorithm) that determines if a given speech recording is bonafide or a fake recording (spoofed). As for the performance evaluation, the equal error rate (EER) was used as a primary metric in the 2015 and 2017 edition. As for the 2019 edition, a recently introduced tandem detection cost function (t-DCF) metric [Kinnunen et al., 2018] was used as a primary metric and EER as the secondary metric. Section 3.5 provides details on these metrics.

As this thesis focusses on replay attacks, we refer readers to [Wu et al., 2015c, 2017] for additional details on the ASVspoof 2015 challenge and results. Subsection 4.3.1 provides a summary of the top performing systems of the ASVspoof 2017 challenge and we refer the reader to [Kinnunen et al., 2017a] for more details. For the 2019 evaluations and the results please refer to [ASVspoof 2019 evaluation_plan, Todisco et al., 2019.].

2.4 Countermeasures for replay spoofing attacks

As defined in Section 1.1, a spoofing countermeasure comprises a frontend feature extractor and a backend classifier. This section now provides a detailed background on different features and modelling approaches that have been studied in the literature for replay spoofing detection. In order to increase the readability, we organize this literature in two Subsections 2.4.1 and 2.4.2. Subsection 2.4.1 groups related works that use traditional hand-designed features and machine learning techniques. Subsection 2.4.2 contains a summary of published literature that uses deep learning either for feature extraction (and uses shallow backend classifiers such as GMMs, SVMs), for classification as a backend model (using hand-crafted features or time-frequency representations of audio as its input) and end-to-end learning where features and classifiers are learned jointly from the training data.

2.4.1 Traditional methods

One of the earliest works studying the impact of playing back recorded speech to fool an ASV system was by [Lindberg and Blomberg, 1999]. This study was performed in a text-dependent setting using a Swedish database for telephone speaker verification. They concatenated pre-recorded digits and simulated replay attacks demonstrating increased false acceptance rates of the system for both male and female speakers. Replay attacks in a remote telephonic application setting were studied in [Shang and Stevenson, 2010]. If a test utterance was found to be very similar to the ones present in the database, then it was considered a replay attack as no speaker can reproduce the exact same utterance.

The study by [Lipeng et al., 2008] used differences in channel characteristics as a cue for replay detection. They first trained a bonafide channel model using bonafide audio recordings of mute voices. During testing, a test speech recording is considered replayed if its channel characteristics are found to be different than the channel model. Wang et al. [2011] also used a similar idea for detecting replay attack. The recording devices and playback devices induce different channel noise in a replayed speech which would be different for bonafide speech recordings. Using this idea, they used support vector machines to model the channel noise difference between bonafide and replayed speech recordings and demonstrated a reduction in the error rate of the GMM-UBM ASV system under replay attack. The vulnerabilities of replay attacks in a far-field recorded speech setting were studied in [Villalba and Lleida, 2010]. The authors demonstrated an increased error rate of a joint factor analysis (JFA) based ASV system when these recorded speech samples were replayed.

One of the first studies on spoofing detection (including replay attack) us-

ing a publicly available spoofing corpus was by [Ergünay et al., 2015]. The dataset, called AVspoof (audio-visual spoofing database) which is different from ASVspoof (Section 3 describes these datasets), consisted of spoofed speech derived using replay, TTS and VC attacks. Using two state-of-the-art ASV systems, the authors demonstrated the vulnerability of ASV systems with increased error rates in the face of spoofing attacks.

Since the release of the benchmark anti-spoofing datasets ASVspoof 2015 [Wu et al., 2015c], ASVspoof 2017 [Kinnunen et al., 2017a] and ASVspoof 2019 [Todisco et al., 2019.] as part of the ongoing ASVspoof challenge series (Section 2.3), there has been considerable research on presentation attack detection [Sahidullah et al., 2019], in particular for TTS, VC, and replay attacks. Many anti-spoofing features coupled with a shallow model (such as GMMs or SVMs) have been studied and proposed in the literature. We briefly discuss them here.

Constant Q cepstral coefficients (CQCCs) [Todisco et al., 2017], among other features, have shown state-of-the-art performance on TTS and VC spoofed speech detection tasks on the ASVspoof 2015 dataset [Wu et al., 2015c]. They have been adapted as baseline features in the recent ASVspoof 2017 and ASVspoof 2019 challenges. Further tweaks on CQCCs have been studied in [Yang et al., 2018a] showing some improvement over the standard CQCCs. Following the widespread adoption of CQCCs in spoofing detection, [Tak et al., 2020] have attempted to understand the effectiveness of CQCCs through a subband analysis with a GMM classifier. The work of [Wang et al., 2017] combines different hand-crafted features with the means and variances of CQCCs to derive a high-dimensional utterance level feature representation. The authors use an SVM classifier for class discrimination.

Jelil et al. [2017] proposed anti-spoofing features based on source and instantaneous frequency and used a GMM for class discrimination. *Teager energy operator* (TEO) based spoof detection features have been studied in [Patil et al., 2017]. Speech demodulation features using the TEO and the Hilbert transform with a GMM classifier have been studied in [Kamble et al., 2018]. Motivated from TEO, Kamble and Patil [2018] proposed a variable length energy separation algorithm using instantaneous amplitude as features coupled with a GMM for replay detection. Furthermore, [Kamble and Patil, 2019] have analysed the impact of reverberation noise on replay spoofing detection using Teager energy features with a GMM backend. Features derived from a source-filter vocal tract model and mel-scale relative phase features with a GMM backend have been studied by Li et al. [2018]. Suthokumar et al. [2018] proposed two utterance level features computed from the modulation spectrum (modulation centroid frequency and modulation spectrum energy) and short term frame based features with a GMM backend classifier for replay spoofing detection. Spectral

centroid based frequency modulation features have been proposed using a GMM in [Gunendradasan et al., 2018].

M S and Murthy [2018] proposed the use of decision level feature switching between mel and linear filterbank slope based features, demonstrating promising performance on the ASVspoof 2017 v2.0 dataset. Features based on compressed integrated linear prediction residuals coupled with a GMM were proposed by [Jelil et al., 2018]. Investigation of several different hand-crafted features that showed good results on synthetic and voice converted speech detection was performed by Font et al. [2017]. Although they use the similar set of features that we used in our work in Section 4.2, our objective and feature configuration is different. We used the default feature configuration that was used by [Sahidullah et al., 2015]. Furthermore, both of these works were part of the ASVspoof 2017 challenge submission. Ensemble models combining scores of several different classifiers (GMMs, SVMs) trained on hand-crafted features (CQCCs, MFCCs and PLPs) have been studied in [Ji et al., 2017].

Motivated from models of the human cochlea, features have been proposed for replay spoofing detection in [Patil et al., 2019, Gunendradasan et al., 2019]. Patil et al. [2019] proposed cochlear cepstral features derived from energy Separation-Based Instantaneous Frequency Estimation with a GMM classifier. Gunendradasan et al. [2019] proposed an Adaptive-Q Cochlear Model from which amplitude modulation features were extracted and GMMs were trained for spoofing detection. Another line of work investigating instantaneous frequency for replay detection is that of [Alluri and Vuppala, 2019]. They proposed three instantaneous cepstral features (single frequency cepstral coefficients, zero time windowing cepstral coefficients, and instantaneous frequency cepstral coefficients) with a GMM for spoofing detection. They have only reported the performance of their proposed features on the development set of the ASVspoof 2019 PA dataset, so it is not clear if their proposed features show good generalisation on unseen test conditions.

Replay spoofing detection based on the blind estimation of the magnitude of channel responses has been studied by [Avila et al., 2019]. Wickramasinghe et al. [2019b] proposed features based on spatial differentiation on the filter outputs of a parallel filter bank for spoofing detection using a GMM classifier. They further proposed to use adaptive bandpass filters and derived three different sets of features with a GMM classifier as a replay spoofing countermeasure [Wickramasinghe et al., 2019a].

The impact of spoofing countermeasures across different phonemes has been studied by [Suthokumar et al., 2019] on the ASVspoof 2017 v2.0 dataset. They developed phone-specific GMMs using rectangular frequency cepstral coefficients (explained in Section 2.5) and demonstrated that more discriminative

information is offered by phoneme groups such as fricatives, nasals, stops and pause phonemes. Liu et al. [2019a] have used attention-based adaptive filters to automatically select only those frequency bands/regions that are most discriminative for spoofing detection. Both phase and magnitude information is used along with a GMM classifier for spoofing detection.

Table 2.1 provides a high-level summary of countermeasures for replay attacks using traditional methods.

2.4.2 Deep learning methods

Within the context of spoofing detection, deep learning models have been proposed either for ***feature learning*** [Qian et al., 2016, Lavrentyeva et al., 2017, Nagarsheth et al., 2017, Sriskandaraja et al., 2018, Sailor et al., 2018, Gomez-Alanis et al., 2019b, Chang et al., 2019, You et al., 2019, Gomez-Alanis et al., 2019a], as a ***backend classifier*** [Yang et al., 2018b, Li et al., 2017, 2019b, Lai et al., 2019b, Białobrzeski et al., 2019, Zeinali et al., 2019, Williams and Rownicka, 2019, Alzantot et al., 2019, Cai et al., 2019, Das et al., 2019, Yang et al., 2019b, Lavrentyeva et al., 2019, Jung et al., 2019, You et al., 2019, Bakar and Hanilçi, 2018, Lai et al., 2019a, Jung et al., 2020, Shim et al., 2019, Yang et al., 2019a], or in an ***end-to-end*** setting to model raw audio waveforms directly [Dinkel et al., 2017, Muckenhirn et al., 2017b].

Multi-task learning (MTL) [Caruana, 1997] and *transfer learning* frameworks along with *attention models* [Goodfellow et al., 2016] have also been explored for spoofing detection tasks. MTL is a branch of machine learning that aims at learning more than one tasks in parallel for improved model generalisation, and it has been investigated in [Li et al., 2019b, Białobrzeski et al., 2019, Platen et al., 2020, Jung et al., 2020, Yang et al., 2019a, Chang et al., 2019] for replay spoofing detection. Transfer learning is a machine learning method that uses the parameters (weights) of a well-trained model for a particular task to initialise a model for a target task. In such a setting either all the model parameters are updated, or only the last few layers are updated while freezing the initial layers of the model (the process is often called finetuning the model) [Goodfellow et al., 2016]. *Transfer learning* and *data augmentation* approaches for spoofing detection have been investigated in [Chang et al., 2019, Shim et al., 2019, Cai et al., 2019] and [Białobrzeski et al., 2019, Yang et al., 2019b] respectively. Attention models on the other hand refer to machine learning techniques that are designed to focus more on those parts of the input that are more relevant and useful for solving a problem [Goodfellow et al., 2016]. Some of the works in spoofing detection using attention models include [Tom et al., 2018] and [Lai et al., 2019a].

Table 2.1: Summary of countermeasures for replay spoofing detection using *traditional methods*. Reported performance is on the evaluation set of the ASVspoof 2017 v1.0 and v2.0 dataset. Ensemble: GMM, SVM, GMM-UBM, GPLDA, Random forest.

Input features	**Classifier**	**Metric (EER%)**	**Dataset**	**Fusion**	**Reference**
Embedding	GMM	7.37	v1.0	-	Lavrentyeva et al. [2017]
CQCC (means+variance) + Emotion feature + MFCC	SVM	24.77	v1.0	-	Wang et al. [2017]
Instantaneous Frequency Cosine Coefficient	GMM	35.19	v1.0	-	Jelil et al. [2017]
Epoch feature	GMM	36.29	v1.0	-	Jelil et al. [2017]
Peak to Side Lobe Ratio mean	GMM	31.60	v1.0	-	Jelil et al. [2017]
CQCC	GMM	19.58	v1.0	-	Jelil et al. [2017]
MFCC	GMM	23.55	v1.0	-	Jelil et al. [2017]
CFCCIF + VESA-IFCC + Prosody	GMM	18.33	v1.0	-	Patil et al. [2017]
LFCCs	GMM	26.27	v1.0	-	Font et al. [2017]
IMFCCs	GMM	30.91	v1.0	-	Font et al. [2017]
RFCCs	GMM	11.9	v1.0	-	Font et al. [2017]
LPCCs	GMM	25.2	v1.0	-	Font et al. [2017]
SCFCs	GMM	24.83	v1.0	-	Font et al. [2017]
SCMCs	GMM	11.49	v1.0	-	Font et al. [2017]
SSFCs	GMM	22.38	v1.0	-	Font et al. [2017]
CQCCs + MFCCs + PLPs	Ensemble	12.4	v1.0	score + feature	Ji et al. [2017]
CQCCs	GMM	13.74	v2.0	-	Delgado et al. [2018]
CQCC i-vector	Cosine	14.76	v2.0	-	Delgado et al. [2018]
Extended CQCCs	GMM	13.38	v2.0	-	Yang et al. [2018a]
ESA-IACC + ESA-IFCC	GMM	9.64	v2.0	-	Kamble et al. [2018]
VESA-IACC	GMM	11.94	v2.0	-	Kamble and Patil [2018]
VESA-IFCC	GMM	11.79	v2.0	-	Kamble and Patil [2018]
VESA-IACC + VESA-IFCC	GMM	7.11	v2.0	-	Kamble and Patil [2018]
Teager Energy Cepstral Coefficients (TECC)	GMM	11.73	v2.0	-	Kamble and Patil [2019]
TECC+CQCC	GMM	11.56	v2.0	-	Kamble and Patil [2019]
TECC+MFCC	GMM	11.73	v2.0	-	Kamble and Patil [2019]
TECC+LFCC	GMM	10.30	v2.0	-	Kamble and Patil [2019]
Phase feature: PBSFVT	GMM	26.58	v2.0	-	Li et al. [2018]
Phase feature: MelRP	GMM	16.03	v2.0	-	Li et al. [2018]
CQCC + PBSFVT + MelRP	GMM	12.88	v2.0	score-level	Li et al. [2018]
Modulation Centroid Frequency	GMM	12.92	v1.0	-	Suthokumar et al. [2018]
Modulation Spectrum Energy	GMM	11.97	v1.0	-	Suthokumar et al. [2018]
Short Term Cepstral Coefficients	GMM	11.27	v1.0	-	Suthokumar et al. [2018]
Spectral Centroid Deviation	GMM	11.45	v1.0	-	Gunendradasan et al. [2018]
Spectral Centroid Frequency	GMM	12.34	v1.0	-	Gunendradasan et al. [2018]
MFCC/LFS/MFS Feature Switching	GMM	6.23	v2.0	-	M S and Murthy [2018]
Compressed Integrated Linear Prediction Residual	GMM	15.76	v2.0	-	Jelil et al. [2018]
CFCCIF-ESA	GMM	14.77	v2.0	-	Patil et al. [2019]
Adaptive-Q Cochlear Amplitude Modulation	GMM	8.09	v2.0	-	Gunendradasan et al. [2019]
MFCC-RASTA	GMM	11.28	v1.0	-	Avila et al. [2019]
Spectral Envelope Centroid Magnitude (CM)	GMM	9.42	v2.0	-	Wickramasinghe et al. [2019a]
CM + Centroid Frequency	GMM	8.58	v2.0	feature	Wickramasinghe et al. [2019b]

Furthermore, some of the well known deep architectures from computer vision such as *ResNet* [He et al., 2015a] and *light CNN* (LCNN) [Wu et al., 2015a] have been widely adopted in spoofing detection demonstrating promising performance on the ASVspoof challenge datasets. For example, the *ResNet* model has been used in [Chen et al., 2017, Cai et al., 2017, Tom et al., 2018, Lai et al., 2019b, Alzantot et al., 2019, Cai et al., 2019, Yang et al., 2019b, Jung et al., 2019, Lai et al., 2019a, Platen et al., 2020, Shim et al., 2019] and, the *light CNN* model was explored in [Lavrentyeva et al., 2017, Białobrzeski et al., 2019, Zeinali et al., 2019, Gomez-Alanis et al., 2019b, Lavrentyeva et al., 2019].

The recently proposed *SincNet* [Ravanelli and Bengio, 2018] architecture for speaker recognition was used for spoofing detection in [Zeinali et al., 2019]. Similarly, x-vectors [Snyder et al., 2018] which were originally proposed for speaker recognition, have been studied for spoofing detection in [Williams and Rownicka, 2019]. Furthermore, *attention-based models* have been studied by Lai et al. [2019a] and Tom et al. [2018] on the ASVspoof 2017 and ASVspoof 2019 datasets respectively.

Lavrentyeva et al. [2017] used the LCNN model on time-frequency representation (spectrogram) inputs to learn discriminative features on which GMM models were trained for spoofing detection. This model demonstrated the best performance as a stand-alone system in the ASVspoof 2017 challenge. Section 4.3 provides further details on this model and the challenges associated in reproducing the model. Nagarsheth et al. [2017] trained a deep CNN using tandem features (concatenation of CQCCs and high frequency cepstral coefficients) to learn different spoofing attack configurations in the training set. Using the trained network, they extract embeddings and an SVM classifier is trained for spoofing detection. Sriskandaraja et al. [2018] trained deep Siamese neural networks (CNNs) that take as input a pair of speech utterances and produces a similarity score, indicating whether the two inputs originate from the same class. They use the network to extract features on which GMMs are trained for spoofing detection. Sailor et al. [2018] proposed the use of convolutional *restricted Boltzmann machines* (RBMs) to learn temporal modulation features for spoofing detection. Gomez-Alanis et al. [2019b] used light convolutional gated recurrent neural networks as deep feature extractors to robustly represent speech signals as utterance-level embeddings and shallow models such as SVMs, linear discriminant analysis (LDA) and probabilistic linear discriminant analysis (PLDA) were used for spoofing detection.

Chang et al. [2019] explored representation learning and transfer learning methods to train a DNN for embedding learning and SVMs are trained on these embeddings for spoofing detection. You et al. [2019] proposed a replay device feature (RDF) extractor on the basis of the genuine-replay-pair training

database in the Constant-Q-Transform (CQT) domain. The RDF feature extractor is proposed based on the CQT transform and Bi-LSTM neural networks to model device specific properties for efficient replay attack detection. The DNN classifier is then trained on RDF features for spoofing detection. This work was performed on the ASVspoof 2017 v2.0 dataset which has balanced training examples between the bonafide and spoof classes. Gomez-Alanis et al. [2019a] used gated recurrent convolutional neural networks as a feature extractor to derive utterance-level embeddings which are then used to train a spoofing detector using shallow models (GMMs and SVMs).

Using the Constant-Q-Transform, Yang et al. [2018b] proposed different types of features such as short-term spectral statistics information, octave-band principal information and fullband principal information. They trained a DNN on the individual features and fused feature (combining all three) for replay spoofing detection, and showed improved performance with the fused features. The influence of various factors such as speaker identity, speech content and playback & recording device towards model overfitting have been studied in [Li et al., 2017]. The authors proposed an F-ratio probing tool for this, and GMMs, SVMs and DNN classifiers were investigated. Their analysis showed that device is the most influential factor contributing towards the risk of overfitting. [Li et al., 2019b] proposed a framework using time delay neural networks that integrates multiple types of features (MFCCs, CQCCs, filter banks, spectrograms) and trains the network in a multi-task setting for spoofing detection.

Using CQCCs and log power spectrogram features [Lai et al., 2019b] trained Squeeze-Excitation Network [Hu et al., 2018], ResNet and their variants in a multi-class setting for spoofing detection. Bayesian neural networks (BNNs) for spoofing detection have been studied by [Białobrzeski et al., 2019]. Along with BNNs they also investigated the LCNN model on spectrogram inputs. Using data augmentation they trained these models in a multi-tasking setting for improved generalisation and model robustness. Zeinali et al. [2019] investigated the use of VGG and LCNN model architectures for spoofing detection on the ASVspoof 2019 PA dataset using MFCCs, CQCCs and the spectrogram as input features. Williams and Rownicka [2019] used SCMC features concatenated with x-vector attack embedding to train a CNN for spoofing detection (ASVspoof 2019 PA dataset). Alzantot et al. [2019] investigated three different variants of a residual convolutional neural network using different feature representations such as MFCCs, spectrogram and CQCCs for replay spoofing detection. ResNet models utilizing various input features such as IMFCCs, CQCCs, power spectrograms along with data augmentation have been studied for spoofing detection in [Cai et al., 2019].

Different variants of CQCC features were investigated by [Das et al., 2019]

using GMM and DNN backends for spoofing detection. Yang et al. [2019b] used Log-constant Q transform (CQT) and a latent vector from a pretrained variational autoencoder as input features to train a spoofing detector that used ResNet and LCNN architectures. Lavrentyeva et al. [2019] proposed changes in their original LCNN architecture that was used in the 2017 evaluations [Lavrentyeva et al., 2017]. In this work they used the LCNN model in an end to end setting where they used output activations as a score, unlike prior work in [Lavrentyeva et al., 2017] where they trained GMMs on top of learned features from the LCNN model. Furthermore, they proposed the angular margin based softmax activation for training a robust deep LCNN classifier in [Lavrentyeva et al., 2019]. On both the tasks of the ASVspoof 2019 challenge their proposed model secured the second rank.

Jung et al. [2019] investigated ResNet and CNN-gated recurrent neural networks for spoofing detection using high resolution magnitude spectrogram (2048 FFT bins) inputs. The influence of complementary information such as phase information and power spectral density on the detection performance was also studied. Bakar and Hanilçi [2018] used long term average spectrum features to train a DNN for replay spoofing detection and suggested that high-frequency components convey more discriminative information. Lai et al. [2019a] used an attention mechanism-based filter that enhances discriminative features prior to a ResNet-based classifier for spoofing detection.

Self-supervised learning approaches leveraging freely-available out-of-domain data (audio recordings from YouTube) for pretraining a DNN to learn acoustic configurations have been investigated in a recent paper by [Shim et al., 2019]. Their proposed framework operates on two stages. First, a DNN is trained to learn acoustic configurations by optimizing the model to detect the similarity between a given pair of segments (which is extracted initially by splicing the original utterances into different segments). Second, they apply transfer learning using the pre-trained DNN weights on the spoofing detection tasks demonstrating improved performance.

Motivated from the successful application of CQT, [Yang et al., 2019a] proposed a multi-level transform framework using DNNs to capture relevant information for spoofing detection by exploiting the octave power spectra of long-term CQT. Tom et al. [2018] used a pre-trained ResNet model (originally trained on images) using an attention mechanism to learn a feature mask from the input which is a group-delay feature matrix. Then, they apply the learned feature mask on the original group delay feature matrix and the pre-trained ResNet model is finetuned for the spoofing detection task. Interestingly, their model demonstrates 0% EER on both the development and evaluation sets of the ASVspoof 2017 v1.0 dataset. Wickramasinghe et al. [2018] proposed to use

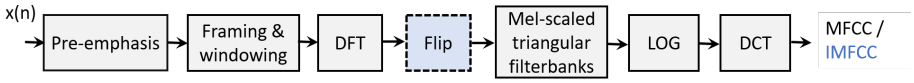


Figure 2.4: MFCC/IMFCC feature extraction pipeline. The flip block applies to IMFCC only.

long-term temporal envelopes of subband signals using a frequency domain linear prediction (FDLP) framework for spoofing detection. Domain adversarial training for spoofing detection has been studied in [Wang et al., 2019a] using an adapted version of the LCNN model architecture.

Using a multi-task DNN framework, Jung et al. [2020] attempts to understand the impact of various factors such as 'Room Size', 'Reverberation', 'Speaker-to-ASV distance, 'Attacker-to-Speaker distance', and 'Replay Device Quality' on replay spoofing detection (on the ASVspoof 2019 PA dataset). Furthermore, another line of work utilizing a multi-task learning framework and Siamese Neural Networks for replay spoofing detection is that of [Platen et al., 2020]. They investigated training this framework on a variety of input features such as log filter bank outputs and group delay matrix.

It is also worth noting that the best performing models on the ASVspoof challenges used *fusion* approaches, either at the classifier output or the feature level [Lai et al., 2019b, Lavrentyeva et al., 2019, 2017, Lai et al., 2019b], indicating the challenges in designing a single countermeasure capable of capturing all the variabilities that may appear in wild test conditions in a presentation attack. Table 2.2 provides a summary of *deep learning* based countermeasures for replay spoofing attack detection. As Table 2.2 summarises, there is a substantial body of prior work on deep models in ASV anti-spoofing, even if it is hard to pinpoint commonly-adopted or outstanding methods. Nonetheless, the majority of the approaches rely either on discriminative models or on classical (shallow) generative models.

2.5 Signal processing methods

This section briefly describes signal processing methods that showed good performance on the detection of TTS and VC spoofing attacks [Sahidullah et al., 2015]. These methods have been applied in this thesis for replay spoofing detection to study how well they generalise on acoustically different attack conditions.

Mel-frequency cepstral coefficients (MFCCs). MFCCs are popular frontend features designed initially for automatic speech recognition [Davis and Mermelstein, 1980] and adopted for speaker recognition applications [Kinnunen and

Table 2.2: Summary of *deep learning* methods for spoofing detection in ASV. Disc: discriminative, Gen: generative. D1: ASVspoof 2015, D2: AVspoof 2016, D3.1: ASVspoof 2017 v1.0, D3.2: ASVspoof 2017 v2.0, D4: ASVspoof 2019 LA, D5: ASVspoof 2019 PA. Details on these datasets and metrics are provided in Chapter 3. Reported numbers are for the respective evaluation sets.

Modeling approach	**Model/ Architecture**	**Input features**	**Purpose**	**Metric [EER/t-DCF]**	**Dataset used**	**Fusion**	**anti-spoofing reference**
Disc	CNN	raw-waveform	end-to-end	0.157	D1	-	Muckenhirn et al. [2017b]
Disc	CNN+LSTM	raw-waveform	end-to-end	[0.82 HTER/-]	D2	-	Dinkel et al. [2017]
Disc	CNN	spectrogram	classification	[10.6/-]	D3.2	-	Section 4.5.2
Disc	LCNN [Wu et al., 2015a]	spectrogram	embedding learning	-	D3.1	-	Lavrentyeva et al. [2017]
Gen	GMMs	embedding	classification	[7.37/-]	D3.1	-	Lavrentyeva et al. [2017]
Disc	CNN	CQCC+HFCC	embedding learning	[11.5/-]	D3.1	feature	Nagarsheth et al. [2017]
Disc	DNN+RNN	FBanks, MFCCs	embedding learning	-	D1	-	Qian et al. [2016]
Disc	SVM, LDA	embedding	classification	[1.1/-]	D1	score	Qian et al. [2016]
Disc	CNN		embedding learning	-	D3.1	-	Sriskandaraja et al. [2018]
Gen	GMM	embedding	classification	[6.4/-]	D3.1	-	Sriskandaraja et al. [2018]
Disc	Bayesian DNN, LCNN	spectrogram	classification	[0.88/0.0219]	D5	score	Białobrzeski et al. [2019]
Disc	VGG, SincNet, LCNN	spectrogram, CQT	classification	[1.51/0.0372]	D5	score	Zeinali et al. [2019]
				[8.01/0.2080]	D4	score	Zeinali et al. [2019]
Disc	LCNN, RNN	spectrogram	embedding learning	-	-	-	Gomez-Alanis et al. [2019b]
Gen	PLDA	embedding	classification	[6.08/-]	D3.1	-	Gomez-Alanis et al. [2019b]
Disc	LDA	embedding	classification	[6.28/0.1523]	D4	-	Gomez-Alanis et al. [2019b]
Gen	PLDA	embedding	classification	[2.23/0.0614]	D5	-	Gomez-Alanis et al. [2019b]
Disc	TDNN, LCNN, ResNet	CQCCs, LFCCs	embedding learning	[9.08/0.1791]	D4	score	Chang et al. [2019]
Disc	ResNet [He et al., 2015a]	STFT, group delay gram	classification	[0.66/0.0168]	D5	score	Cai et al. [2019]
Disc	TDNN	MFCC, CQCC, spectrogram	multitasking, classification	[7.94/-]	D3.2	score	Li et al. [2019b]
				[7.63/0.2129]	D4	score	Li et al. [2019b]
				[0.96/0.0266]	D5	score	Li et al. [2019b]
Disc	ResNet	MFCCs, CQCCs	classification	[13.30/-]	D3.1	score	Chen et al. [2017]
Disc	ResNet, SeNet	CQCC, spectrogram	classification	[0.59/0.016]	D5	score	Lai et al. [2019b]
				[6.7/0.155]	D4	score	Lai et al. [2019b]
Disc	ResNet	MFCCs, CQCC, spectrogram	classification	[6.02/0.1569]	D4	score	Alzantot et al. [2019]
				[2.78/0.0693]	D5	score	Alzantot et al. [2019]
Disc	CNN+GRU	spectrogram	classification	[2.45/0.0570]	D5	score	Jung et al. [2019]
	ResNet, LSTMs	spectrogram	embedding learning	[16.39/-]	D3.1	score	Cai et al. [2017]
Gen	GMMs	CQCC, LFCC, MelRP	classification	[11.43/-]	D3.2	score	Liu et al. [2019a]
Gen	Attention-based ResNet	spectrogram	classification	[8.54/-]	D3.2	score	Lai et al. [2019a]
Gen	VAE	CQT spectrogram	embedding learning	-	D4	-	Yang et al. [2019b]
Gen	VAE	CQCC, spectrogram	classification, feature extraction		D3.2, D5		Section 5.5

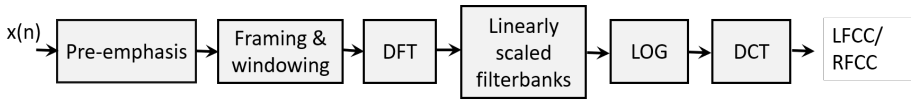


Figure 2.5: LFCC/RFCC feature extraction pipeline. LFCCs use triangular filters and RFCCs use rectangular filters.

Li, 2010a]. These features are based on a model of human auditory perception. Fig. 2.4 summarises the steps involved in extracting this feature. A pre-emphasis filter is first applied to the speech signal to boost energy in the higher frequency components. This is followed by slicing the speech signal into overlapping frames through framing and windowing operations. As for windowing, Hanning or Hamming windows are generally used to smooth the edges (boundary after the split). The discrete Fourier transform (DFT) [Dickinson and Steiglitz, 1982] is then applied on the windowed frames to extract information in the frequency domain. The output from the DFT (magnitude spectrum) is first squared and Mel scaled triangular bandpass filters are applied on DFT power spectrum. The main motivation of using Mel filters is to mimic how humans perceive sounds. These filters are linearly spaced below 1000 Hz and are spaced logarithmically above this frequency [Lerch, 2012]. Furthermore, they give less emphasis on higher frequency components (widely spaced filters) to reflect human hearing sensitivity at higher frequencies. Next, a log compression on the mel-scaled power spectrum is applied. The final step involves taking the discrete cosine transformation to decorrelate the mel-spectral feature vectors yielding the desired MFCCs. Usually the first few coefficients are retained as a feature representation per frame.

Inverted Mel-frequency cepstral coefficients (IMFCCs). IMFCCs were first introduced for speaker recognition applications by Chakroborty et al. [2007], and they have been successfully applied for detecting synthetic and voice converted speech [Sahidullah et al., 2015]. The feature extraction pipeline of IMFCCs is the same as in MFCCs, as depicted in Fig. 2.4 with one key difference. Unlike MFCCs which put higher emphasis in lower frequency regions, the mel-scaled filters are flipped to emphasise more on the higher frequency components. In other words, mel-scaled filters are now linearly spaced above 7000 Hz frequencies and are spaced logartihmically below this frequency. In practise, this effect can be incorporated by simply adding a flip block, as highlighted in blue in Fig. 2.4, that inverts the output of the DFT before applying Mel scaled triangular bandpass filters on it.

Linear frequency cepstral coefficients (LFCCs). LFCC frontends, initially intro-

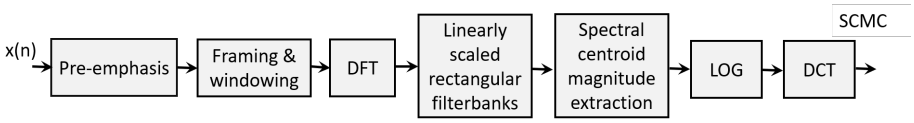


Figure 2.6: SCMC feature extraction pipeline.

Figure 2.7: CQCC feature extraction pipeline.

duced for speaker recognition, have been widely adopted in spoofing detection reporting a good performance [Sahidullah et al., 2015]. Fig. 2.5 illustrates the pipeline for LFCC feature extraction, which is similar to that of the standard MFCCs with one key difference. Here the filters are placed in equal sizes following a linear scale unlike MFCCs that use mel scale spacing.

Rectangular frequency cepstral coefficients (RFCCs). The RFCC features have shown good results in detecting artificial speech produced through TTS and VC algorithms [Sahidullah et al., 2015]. The steps involved in extracting RFCCs are very similar to that of LFCCs with the key difference being the filter shape. RFCCs as the name suggests use rectangular shaped filters spaced at a linear scale. Fig. 2.5 summarises the RFCC feature extraction pipeline.

Furthermore, Fig. 2.8 provides a visual summary of different filters used during the extraction of the hand-crafted features discussed so far.

Subband centroid magnitude coefficients (SCMCs). SCMC [Min Karen Kua et al., 2010] features were first introduced by Sahidullah et al. [2015] for spoofing detection demonstrating a good detection performance for TTS and VC spoofing attacks. Fig. 2.6 illustrates the steps for extracting SCMC features. The first four steps are the same as used in computing RFCCs. This is followed by the computation of the spectral centroid magnitude from the magnitude spectrum, which is defined as the weighted (frequency of each magnitude component is used as weights) average magnitude for a given subband. This step is followed by the standard log non-linearity and DCT giving the desired SCMC features.

Constant Q cepstral coefficients (CQCCs). This feature was proposed by Todisco et al. [2016, 2017] for TTS and VC spoofing detection and is now widely adopted as a default spoofing detection feature following its good performance on these

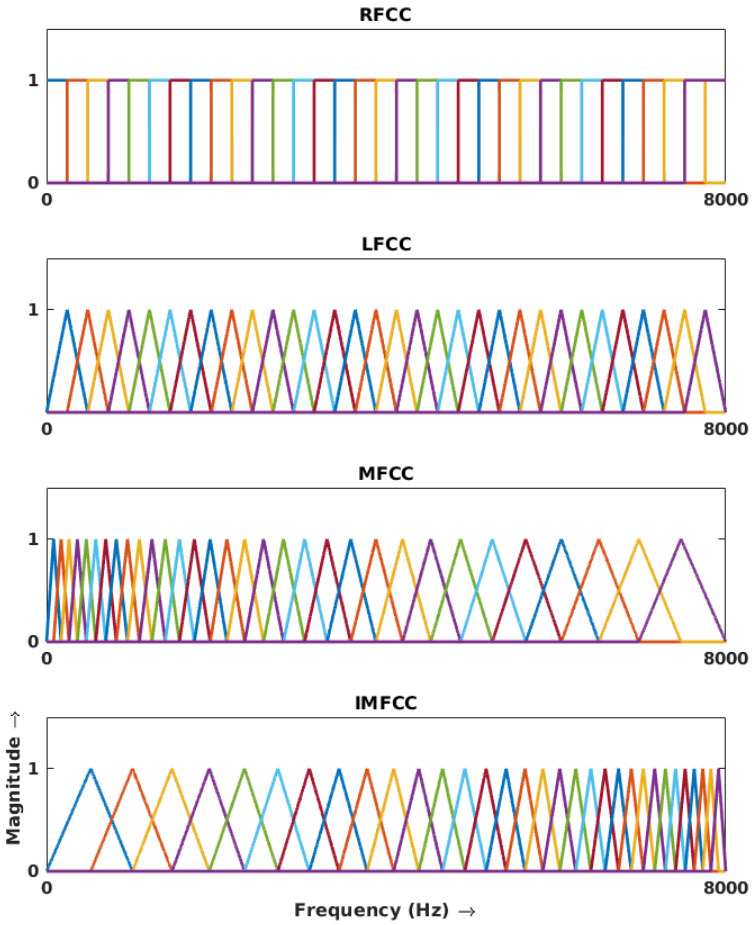


Figure 2.8: Illustration of filters used in extracting RFCC, LFCC, MFCC and IMFCC features.

tasks. Unlike other cepstral features discussed so far which use the short time Fourier transform (STFT), the CQCC feature extraction process uses a constant Q transform (CQT) which provides variable time and frequency resolution. It provides higher frequency resolution at lower frequencies and higher time resolution at higher frequencies in contrast to STFT that use fixed time and frequency resolutions. Fig. 2.7 summarises the steps involved in extracting CQCC features. The CQT spectrum is first obtained from the given speech signal from which the power spectrum is computed. This is followed by the application of a log non-linearity. A uniform resampling operation is then applied to convert the non-uniform frequency scale of CQT to a linear frequency scale, followed by the conventional DCT operation resulting in the desired CQCC features.

All of the above described hand-crafted features have been used in Section 4.2. Furthermore, CQCCs have also been studied in Sections 4.4, 4.6, 5.3, and 5.5. MFCCs and IMFCCs have also been studied in Sections 4.4.

2.6 Discriminative models

This section provides a description of discriminative models that have been used as a backend classifier for spoofing detection in the literature and in this thesis.

2.6.1 Support vector machine

The support vector machine (SVM) is a supervised learning algorithm that aims at solving classification problems, and is well-known for binary classification problems. Using kernel trick [Campbell et al., 2006], it maps the original input into a high-dimensional space to learn a decision boundary (hyperplane) separating the two classes while maximizing the margin of separation between the two classes. Such a decision boundary is often referred to as the optimal hyperplane and the data points closest to this hyperplane are called support vectors.

SVMs were one of the popular classifiers used in speaker recognition [Campbell et al., 2006, 2007]. It has been adopted for spoofing detection problems recently [Sahidullah et al., 2015, Delgado et al., 2018]. Training an SVM for spoofing detection requires labelled training examples for both the bonafide and spoof classes. Usually, bonafide classes are labelled with a +1 and -1 is used for the spoof class. In the context of speaker verification, +1 corresponds to a target speaker and -1 to the background speaker (representing the world population). From these labelled training features, the optimizer aims to learn a separating hyperplane in the high-dimensional space by maximizing the margin of separation between the two classes. The classifier discriminant function

[Campbell et al., 2006] is written as

$$f(x) = \sum_{i=1}^{N} \alpha_i t_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b \tag{2.1}$$

where $\boldsymbol{x}_i$ are the support vectors, t_i corresponds to the class output labels that could be either +1 or -1, $\sum_{i=1}^{N} \alpha_i t_i = 0$, and $\alpha_i > 0$. These parameters along with the bias term b are estimated from the training data through some optimisation process [Campbell et al., 2006]. Here the kernel function $K(.,.)$ is defined as a mapping of the input feature space into a high dimensional kernel space, expressed as

$$K(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^{\mathrm{T}} \phi(\boldsymbol{y}) \tag{2.2}$$

where $\phi(.)$ represents a mapping from the input space to a high dimensional space. The non-linear data in the input space that was difficult to classify becomes linearly separable in a higher dimensional space and hence classification becomes easier.

2.6.2 Convolutional neural networks

A CNN is a class of neural network architecture that was primarily designed for visual recognition applications [LeCun et al., 1999], but, recently they have been successfully adopted in various audio and speech technology applications. Examples include ASR [Huang et al., 2019], acoustic scene classification [Samarth R Phaye et al., 2019], sound event detection [Chan et al., 2020], language recognition [Bartz et al., 2017], and speaker recognition [Muckenhirn et al., 2018]. CNNs have also been widely studied for spoofing detection [Zeinali et al., 2019, Gomez-Alanis et al., 2019b, Lavrentyeva et al., 2017, Nagarsheth et al., 2017]. Dominant systems in the ASVspoof challenges used CNNs as standalone systems or in ensemble setting producing impressive results [Lavrentyeva et al., 2017, Lai et al., 2019b].

CNNs follow the concept of weight sharing, where all the neurons in a particular feature map share the same weight parameters. Another important concept is local connectivity, where not every neuron in the previous layer is connected to every neuron in the following layer. A neuron is connected to only a small subset of the input representation. This helps control the trainable parameters of the network and makes it computationally efficient which otherwise would be difficult if fully connected (FC) networks were used to train on high-dimensional image data.

Unlike feed forward or fully connected neural networks where every neuron in any given layer (except the input layer) receives inputs from every unit of the

previous layer, CNNs follow the concept of weight sharing where every neuron does not receive the whole inputs from the previous layer, rather it only sees a proportion of the input. The main motivation for this design is two-fold. First, to control the number of trainable parameters (which otherwise would be enormous when high dimensional image data were flattened into a single vector and trained using a FC network). The second motivation comes from the way receptive fields work in the visual cortex [LeCun et al., 1999]. A CNN contains the following building blocks: convolutional, activations (non-linearity), pooling (sub sampling), and classification.

Convolutional (Conv) layer is the main building block of CNNs. It takes in an input signal and applies a filter over it. In other words, it performs a dot product between the input and a kernel (filters) to obtain the convolved or modified signal. This layer comprises a set of independent kernels or filters initialised randomly and is learned during training. Each kernel/filter is independently convolved with the input image producing a set of feature maps.

Activation functions are mainly used to introduce non-linearity in the learning process enabling models to capture non-linear relationships in the data. Although there exist several activation functions [Goodfellow et al., 2016] in the literature that are used in training deep models, we provide a brief review of two of these activations that are directly relevant to this thesis.

- *Rectified linear unit (ReLU)* [Nair and Hinton, 2010]. It is a widely used activation function in training deep models and is often used as a default activation function. It is defined as:

$$f(x) = max(0, x) \tag{2.3}$$

  For every input x, ReLU performs a simple thresholding operation allowing positive values to pass through while replacing negative values with a 0. A number of changes were introduced on ReLU and new activation functions have been proposed in the literature. These include the exponential linear unit (ELU) [Clevert et al., 2015], leaky ReLU [Maas et al., 2013], parametric rectified linear unit (PReLU) [He et al., 2015b], and scaled exponential linear unit (SeLU) [Klambauer et al., 2017].

- *Max-Feature-Map (MFM).* MFM non-linearity was originally introduced by Wu et al. [2015a] to train a deep neural network called Light CNN (LCNN) for face recognition. The authors claim that the MFM design was motivated to help select optimal feature maps during model training.

It is defined as:

$$\begin{aligned} y_{ij}^{k} &= max(x_{ij}^{k}, x_{ij}^{k+\frac{N}{2}}) \\ \forall i &= \overline{1,H}, j = \overline{1,W}, k = \overline{1,N/2}, \end{aligned} \tag{2.4}$$

where x is a 3D input tensor of shape $H \times W \times N$ and y corresponds to output tensor having a shape $H \times W \times \frac{N}{2}$. Here, H, W and W corresponds to height, width and channel depth for a tensor. The LCNN model with MFM activation demonstrated the best performance during the ASVspoof 2017 challenge evaluations [Lavrentyeva et al., 2017]. Section 4.3 provides a summary of the LCNN model used in the challenge.

Pooling layer is an important property that is used to down-sample the original input reducing its dimensionality. This operation is performed by applying a pooling filter of small dimension (usually 2×2) onto a feature map and transforms them to a single scalar either by taking the maximum value or average of these pooling filters. The *classification layer* usually takes the output of the last convolutional layer and performs classification tasks using a series of fully connected layers.

Usually, with CNNs, the Conv layers are considered as feature extractors which aim to learn discriminative features, and the output from the last Conv layer which is a flattened vector is fed to a fully connected layer for the final classification task.

In this thesis, CNNs have been used extensively in all the sections of Chapters 4 and 5, and SVMs have been studied in Sections 4.2, 4.4, 4.6, and 5.3 as a backend classifier for spoofing detection.

2.7 Generative models

This section provides the background on three different generative models: Gaussian mixture models, i-vectors and variational autoencoders, that have been studied for spoofing detection in the literature and in this thesis.

2.7.1 Gaussian mixture model (GMM)

A Gaussian mixture model, denoted by $\mathbf{\Lambda}$, is defined as a weighted sum of C component density functions:

$$p(\boldsymbol{x}|\mathbf{\Lambda}) = \sum_{k=1}^{C} w_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k), \tag{2.5}$$

where C is the number of Gaussian components, w_k is the prior probability or mixture weight of the k^{th} component, and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$ is a d-variate Gaussian

density function with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$, defined as:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\ \boldsymbol{\Sigma}_k^{-1}\ (\boldsymbol{x}-\boldsymbol{\mu}_k)\right\} \tag{2.6}$$

Here, the mixture weights $w_k \geqslant 0$ satisfy the constraints $\sum_{k=1}^{C} w_k = 1$.

Training. Training a GMM involves estimating the model parameters $\boldsymbol{\Lambda} = \{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_\mathbf{k}\}_{k=1}^{C}$ that maximize the average log likelihood of the data. This is performed by applying the *expectation maximization* (EM) algorithm [Bishop, 2006] on a training dataset $\mathcal{X} = \{\boldsymbol{x}_t\}_{t=1}^{T}$. The average log-likelihood of $\mathcal{X}$ with respect to model $\boldsymbol{\Lambda}$ is given as:

$$\mathrm{LL}_{avg}(\mathcal{X}|\boldsymbol{\Lambda}) = \frac{1}{T}\sum_{t=1}^{T}\log p(\mathcal{X}|\boldsymbol{\Lambda}) = \frac{1}{T}\sum_{t=1}^{T}\log\sum_{k=1}^{C} w_k\,\mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2.7}$$

Testing. For a given test utterance X consisting of T feature vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$, the GMM computes a mean log-likelihood score using (2.7). The higher the value, the higher is the probability that $\boldsymbol{\Lambda}$ generated the data.

In the automatic spoofing detection task, two GMMs are usually trained one each for the bonafide and spoof classes. During testing, for a given test utterance X, using (2.7) we compute the average log-likelihood difference as the final score:

$$\mathrm{LLR} = \mathrm{LL}_{avg}(\mathcal{X}|\boldsymbol{\Lambda}_{bona}) - \mathrm{LL}_{avg}(\mathcal{X}|\boldsymbol{\Lambda}_{spoof}) \tag{2.8}$$

where $\boldsymbol{\Lambda}_{bona}$ and $\boldsymbol{\Lambda}_{spoof}$ represents the bonafide and spoof GMMs. The larger the LLR value is, the more confidence the model has that the test utterance X is a bonafide utterance.

2.7.2 i-vectors

An i-vector is a fixed-dimensional utterance-level representation derived from a variable-length speech signal through factor analysis [Dehak et al., 2011]. This representation aims at capturing the long-term characteristics in a speech signal, such as speaker and channel properties. It models both speaker and channel variabilities into a single low-rank space called total variability space, and is defined by the equation:

$$\boldsymbol{s} = \boldsymbol{m} + \mathbf{T}\boldsymbol{\phi} \tag{2.9}$$

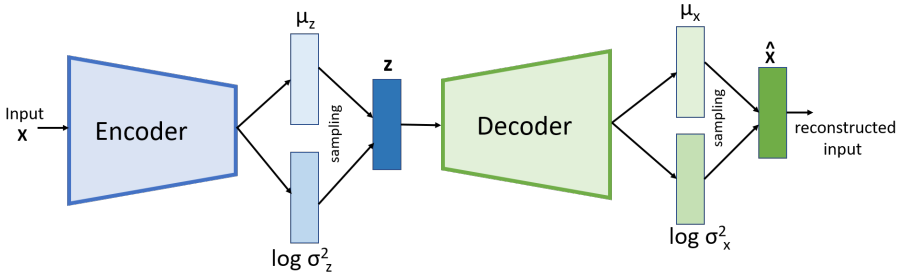


Figure 2.9: Block diagram illustrating a Variational Autoencoder.

where $\boldsymbol{s}$ is a speaker and channel dependent GMM mean supervector[2], $\boldsymbol{m}$ is a UBM supervector which is speaker and channel independent, $\mathbf{T}$ is a rectangular matrix of low rank (total variability space) and $\boldsymbol{\phi}$ is an identity vector, commonly referred to as i-vector.

Initially, i-vectors were proposed for speaker recognition applications and were state-of-the-art features until discriminatively trained deep learning-based x-vectors [Snyder et al., 2018] were introduced recently. Channel compensation methods are often applied on i-vectors to minimize the channel effects before using them for speaker modelling. They have been widely used in other speech technology applications as well. For example, in automatic speech recognition, i-vectors are often provided as an additional input along with other acoustic features (eg. MFCCs) which helps improve the robustness of acoustic models [Gupta et al., 2014]. They have also been recently explored in spoofing detection applications [Novoselov et al., 2016a, Delgado et al., 2018]. For more details on i-vectors please refer to [Dehak et al., 2011] and [Kanagasundaram, 2014]. We used the MSR identity toolkit [Sadjadi et al., 2013] for computing i-vectors.

2.7.3 Variational Autoencoders (VAEs)

Variational Autoencoder (VAE) [Kingma and Welling, 2013] is a *deep generative model* that aims at uncovering the data generation mechanism in the form of a probability distribution. The VAE is an *unsupervised* approach that learns a low-dimensional, nonlinear data *manifold* from training data without class labels. VAEs achieve this by using two separate but jointly trained neural networks, an *encoder* and a *decoder* as illustrated in Fig. 2.9. The encoder forces the input data through a low-dimensional *latent space* that the decoder uses to reconstruct the input.

[2]A supervector is a high dimensional vector obtained by stacking the mean vectors from all the Gaussian components of a UBM. A UBM is essentially a GMM trained on a large speech corpus comprising many thousands of speakers.

Given a D-dimensional input $\boldsymbol{x} \in \mathbb{R}^D$, the encoder network maps $\boldsymbol{x}$ into a latent vector $\boldsymbol{z} \in \mathbb{R}^d$ ($d \ll D$). Unlike in a conventional (deterministic) autoencoder, $\boldsymbol{z}$ is not a single point; instead, the encoder imposes a *distribution* over the latent variable, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, where $\boldsymbol{\phi}$ denotes all the parameters (network weights) of the encoder. The default choice, also in this work, is a Gaussian $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \mathsf{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})\right))$, where $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$ and $\mathsf{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})\right)$ are deterministic functions (the encoder network) that return the mean and variance vector (*i.e.*, diagonal covariance matrix) of the latent space given an input $\boldsymbol{x}$.

The decoder network, in turn, takes $\boldsymbol{z}$ as input and returns a parameterized probability distribution, which is another Gaussian. The decoder distribution is $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \mathsf{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{z})\right))$, where $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{z})$ and $\mathsf{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{z})\right)$ are deterministic functions implemented by the decoder network, and where $\boldsymbol{\theta}$ denotes the decoder network parameters. Random observations sampled from the decoder distribution (with fixed $\boldsymbol{z}$) should then bear resemblance to the input $\boldsymbol{x}$. In the standard VAE, the only sampling that takes place is from the variational posterior distribution of the latent variable. Conceptually, however, it is useful to note that the decoder also produces a distribution of possible outputs, rather than a single point estimate, for a given (fixed) $\boldsymbol{z}$. These outputs will not be exactly the same as $\boldsymbol{x}$ due to the dimensionality reduction to the lower-dimensional $\boldsymbol{z}$-space, but each of the individual elements of the $\boldsymbol{z}$-space represents some salient, meaningful features necessary for approximating $\boldsymbol{x}$.

VAE training. The VAE is trained by maximizing a regularized log-likelihood function. Let $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ denote the training set, with $\boldsymbol{x}_n \in \mathbb{R}^D$. The training loss for the entire training set $\mathcal{X}$,

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n=1}^{N} \ell_n(\boldsymbol{\theta}, \boldsymbol{\phi}), \tag{2.10}$$

decomposes to a sum of data-point specific losses. The loss of the nth training example is a regularized reconstruction loss:

$$\ell_n(\boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{-\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n|\boldsymbol{z})\Big]}_{\text{Reconstruction error}} + \underbrace{\mathrm{KL}\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n) \,\|\, p(\boldsymbol{z})\big)}_{\text{Regularizer}}, \tag{2.11}$$

where $\mathbb{E}[\cdot]$ denotes the expected value and $\mathrm{KL}(\cdot\|\cdot)$ is the *Kullback-Leibler divergence* [Cover and Thomas, 2001] – a measure of difference between two probability distributions. The reconstruction error term demands for an accurate approximation of $\boldsymbol{x}$ while the KL term penalizes the deviation of the encoder distribution from a fixed *prior distribution*, $p(\boldsymbol{z})$. Note that the prior, taken to be the standard normal, $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$, is shared across all the training

exemplars. It enforces the latent variables $\boldsymbol{z}$ to reside in a compatible feature space across the training exemplars.

In practice, to derive a differentiable neural network after sampling $\boldsymbol{z}$, VAEs are trained with the aid of the so-called *reparameterization trick* [Kingma and Welling, 2013]. Thus, sampling $\boldsymbol{z}$ from the posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is performed by computing $\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ is a random vector drawn from $\mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the means and variance of the posterior learned during the VAE training, and $\odot$ denotes the element-wise product.

VAEs and GMMs as latent variable models. Given the widespread use of GMMs in voice anti-spoofing studies, it is useful to compare and contrast the two. Similar to the VAE, the GMM is also a generative model that includes latent variables. In the case of GMMs, $\boldsymbol{x}$ is a short-term feature vector, and $\boldsymbol{z}$ is a one-hot vector with C components (the number of Gaussians), indicating which Gaussian was 'responsible' for generating $\boldsymbol{x}$. Let $\boldsymbol{z}_k = (0, 0, \ldots, 1, 0, \ldots, 0)^{\mathsf{T}}$ be a realization of such one-hot vector where the k-th element is 1. The conditional and prior distributions of GMM are:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{z} = \boldsymbol{z}_k, \boldsymbol{\Lambda}) &= \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ p(\boldsymbol{z} = \boldsymbol{z}_k, \boldsymbol{\Lambda}) &= w_k, \end{aligned} \tag{2.12}$$

where $\boldsymbol{\Lambda} = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, w_k)_{k=1}^{C}$ denotes the GMM parameters (means, covariances and mixing weights). By marginalizing the latent variable out, the log-likelihood function of a GMM as defined in (2.7) is used as a score when comparing test feature $\boldsymbol{x}$ against the GMM defined by $\boldsymbol{\Lambda}$.

Both are generative approaches, and common to both is the assumption of the data generation process consisting of two consecutive steps:

1. **First**, one draws a latent variable $\boldsymbol{z}_n \sim p_{\text{gt}}(\boldsymbol{z})$ from a *prior distribution*.
2. **Second**, given the selected latent variable, one draws the observation from a *conditional distribution*, $\boldsymbol{x}_n \sim p_{\text{gt}}(\boldsymbol{x}|\boldsymbol{z}_n)$,

where the subscript 'gt' highlights an assumed underlying 'true' data generator whose details are unknown. Both VAEs and GMMs use parametric distributions to approximate $p_{\text{gt}}(\boldsymbol{z})$ and $p_{\text{gt}}(\boldsymbol{x}|\boldsymbol{z}_n)$. In terms of the '$\boldsymbol{z}$' variable, the main difference between GMMs and VAEs is that in the former it is discrete (categorical) and in the latter it is continuous. As for the second step, in GMMs, one draws the observation from a multivariate Gaussian distribution corresponding to the selected component. In VAEs, one also samples the reconstructed obser-

vation from a Gaussian, but the mean and covariance are not selected from an enumerable set — they are continuous and are predicted by the decoder from a given $\boldsymbol{z}$.

Both GMMs and VAEs are trained with the aim of finding model parameters that maximize the training data log-likelihood; common to both is that no closed-form solution for the model parameters exists. The way the two models approach the parameter estimation (learning) problem differs substantially, however. As in any maximum likelihood estimation problem, the training observations are assumed to be i.i.d., enabling the log-likelihood function over the whole training dataset to be written as the sum of log-likelihoods over all the training observations. This holds both for VAEs and GMMs. Let us use the GMM as an example. For a single observation $\boldsymbol{x}$, the log-likelihood function is:

$$\begin{aligned}\log p_{\boldsymbol{\Lambda}}(\boldsymbol{x}) = \log \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\Lambda}) = \sum_{\boldsymbol{z}} Q(\boldsymbol{z}) \frac{p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\Lambda})}{Q(\boldsymbol{z})} = \log \ \mathbb{E}_{\boldsymbol{z}\sim Q(\boldsymbol{z})}\left[\frac{p_{\boldsymbol{\Lambda}}(\boldsymbol{x}, \boldsymbol{z})}{Q(\boldsymbol{z})}\right] \\ \geq \mathbb{E}_{\boldsymbol{z}\sim Q(\boldsymbol{z})}\left[\log \frac{p_{\boldsymbol{\Lambda}}(\boldsymbol{x}, \boldsymbol{z})}{Q(\boldsymbol{z})}\right] = \sum_{\boldsymbol{z}} Q(\boldsymbol{z}) \log \frac{p_{\boldsymbol{\Lambda}}(\boldsymbol{x}, \boldsymbol{z})}{Q(\boldsymbol{z})}\end{aligned} \tag{2.13}$$

where $Q(\boldsymbol{z})$ is *any* distribution, and where the inequality in the second line is obtained using *Jensen's inequality* [Cover and Thomas, 2001] (using the concavity of the logarithm). The resulting last expression, known as the *evidence lower bound* (ELBO), serves as a lower bound of the log-likelihood which can be maximized more easily. The well-known EM algorithm [Dempster et al., 1977] is an alternating maximization approach which iterates between updating the Q-distribution and the model parameters $\boldsymbol{\Lambda}$ (keeping the other one fixed when updating the other one). An important characteristic of the EM algorithm is that, in each iteration, the posterior distribution $Q(\boldsymbol{z})$ is selected to make the inequality in (2.13) *tight*, making the ELBO *equal* to the log-likelihood. This is done by choosing $Q(\boldsymbol{z})$ to be the posterior distribution $P_{\boldsymbol{\Lambda}}(\boldsymbol{z}|\boldsymbol{x})$ (using the current estimates of model parameters). Importantly, this posterior can be computed in *closed form*. The EM algorithm is guaranteed to converge to a local maximum of the log-likelihood. It should be noted, however, that as the likelihood function contains local maximae [Jin et al., 2016], global optimality is not guaranteed. The quality of the obtained GMM (in terms of log-likelihood) depends not only on the number of EM iterations, but on the initial parameters.

In contrast to GMMs, the posterior distribution of VAEs *cannot* be evaluated in closed form at any stage (training or scoring). For this reason, it is replaced by an *approximate*, variational [Bishop, 2006] posterior, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, leading to the ELBO training objective of Eq. (2.11). As the true posterior distribution cannot be evaluated, the EM algorithm cannot be used for VAE training [Kingma and

Welling, 2013]. The ELBO is instead optimized using gradient-based methods. Due to all these differences, it is difficult to form an exact comparison between the VAE and GMM models. One of the main benefits of VAEs over GMMs is that they can handle high-dimensional inputs — for example, raw spectrograms and CQCC-grams consisting of multiple stacked frames — allowing the use of less restrictive features.

In this thesis, GMMs have been used in Sections 4.2, 4.4, 4.6, 5.2, 5.3, and 5.5. And, i-vectors have been used in Sections 4.2, 4.4, 4.6, 5.2, and 5.3. VAEs have been studied in Section 5.5.

2.8 Subband modelling

A subband represents a sub-part of the speech signal frequencies, which are usually extracted to perform some specific task. For example, extracting MFCCs for speech recognition application involves use of lower frequency subbands (typically between 100 to 3500 Hz) and discarding other frequencies as they do not carry relevant information for the task in hand. However, for certain applications higher frequency bands may be useful, for example spoofing detection. This section discusses the relevant background on subbands and its application on various speech applications.

The impact of subbands on model performance has been investigated on a wide range of ML tasks in the literature. For instance, Besacier and Bonastre [2000] used a subband approach to extract relevant features, each modelled using Gaussians for speaker verification. Kingma and Ba [2008] investigated the dependencies of different frequency bands and speaker characteristics in a speech signal for speaker verification applications. Recently, Samarth R Phaye et al. [2019] demonstrated improved performance using different subbands for building acoustic scene classification models. They used a *sub-spectrogram*, obtained by cropping a mel-spectrogram at different bands, to train a convolutional neural network (CNN) for learning band-specific features.

In the context of spoofing detection, the most relevant studies include [Sriskandaraja et al., 2016, Witkowski et al., 2017, Garg et al., 2019, Nagarsheth et al., 2017, Lin et al., 2018, Soni et al., 2016]. Sriskandaraja et al. [2016] investigated different subbands to find the most informative bands useful for spoofing detection tasks. Using the Kullback-Leibler divergence (at model-level) and classification-level analysis, they identified 0-1 kHz, 2.5-5.5 kHz and 7-8 kHz as the most informative subbands for their dataset. Features were then extracted from these bands to train a classifier (*Gaussian mixture model — universal background model*), demonstrating improved performance over traditional full-band models. Witkowski et al. [2017] investigated the importance of different sub-

bands for spoofing detection on the ASVspoof 2017 dataset. They extracted five different types of features from these subbands to train a GMM backend classifier. They found the high frequency range of 6-8 kHz to be the most informative. Another similar line of work was performed by Garg et al. [2019] using CQCC and MFCC features for replay spoofing detection on the ASVspoof 2017 dataset, reporting similar findings as in [Witkowski et al., 2017]. The high frequency bands, 6-8 kHz, was found to offer more discriminative information for replay attack detection on this dataset.

Nagarsheth et al. [2017] proposed *high-frequency cepstral coefficient* (HFCC) features extracted from a high-frequency spectrum (above 3.5 kHz). They combined HFCCs with CQCCs and trained a deep neural network as a feature extractor. A support vector machine classifier was trained on deep features, outperforming the baseline GMM models on the ASVspoof 2017 dataset. Another interesting work in subband modelling is by Soni et al. [2016]. They trained a subband autoencoder (SBAE) for feature extraction by restricting the connections between units in the input and the first hidden layer of the encoder. By imposing such constraints they claim that models learn band-specific features useful in spoofing detection, demonstrating a substantial gain in detection performance on the ASVspoof 2015 dataset. Another line of study that investigates subband features for spoofing detection is by [Lin et al., 2018]. Using the subbands that provide discriminative information, the authors design new filters for feature extraction. Experimental results on the ASVspoof 2017 v1.0 dataset indicate that the 0-1 and 7-8 kHz subbands offer the most discriminative information.

To sum up, previous studies indicate that certain frequency subbands are potentially more informative to the detection of spoofing attacks, even though no standardized approach show how that unevenly distributed information across the frequency axis should be utilised. Our current work (Section 5.4) is different from the prior works mentioned above because most of them aim at handcrafting or learning features [Soni et al., 2016] based on the relevance of specific subbands for spoofing detection. However, our work described in Section 5.4 aims to learn band-specific features by discriminatively training a CNN on a spectrogram input for spoofing detection.

2.9 Towards trustworthy countermeasures

Analysing spoofing detection systems is one of the objectives of this thesis. In this direction, this section reviews related works on the trustworthiness of machine learning models for spoofing detection. Firstly, a high level definition of what "trustworthiness" means in this thesis is provided, and a background on

how artefacts and confounding factors in a dataset impact model trustworthiness is explained in Subsection 2.9.1. Then, the next Subsection 2.9.2 explains a method (called SLIME) from the interpretable machine learning literature that is used in this thesis (Section 4.5) to analyse the predictions of a CNN-based countermeasure for replay spoofing detection.

2.9.1 Artefacts and their influence in machine learning

Following the Ethics Guidelines for Trustworthy AI prepared by the High-Level Expert Group on Artificial Intelligence (AI HLEG) of the European Commission AI-HLEG [2020], this thesis considers technical robustness, fairness and accountability as key criteria for any ML countermeasure model to be deemed trustworthy. These criteria suggest that the results produced by a trustworthy countermeasure should be independent of the variables/factors that are sensitive but not related to the actual problem.

The performance of any data-driven ML task highly depends on the training data fed to the learning algorithm. The model learns to make decisions by exploiting the underlying patterns within the training data [Bishop, 2006]. As demonstrated in [Sturm, 2013, Tommasi et al., 2015, Rosset et al., 2010], such models may easily exploit irrelevant cues, artefacts or confounders (if present) during the training optimisation. Unless explicitly accounted for during training and inference, they can introduce biases[3] in model decisions raising questions on their reliability, and often contribute in achieving good results and overestimating the actual performance on a test set. Such issues can occur in a wide range of ML tasks [Sturm, 2013, Tommasi et al., 2015, Rodríguez-Algarra et al., 2019, Rosset et al., 2010, Stowell et al., 2019]. Sturm [2013] shows how faults in GTZAN, a popular dataset for musical genre classification, overestimated classification accuracy. Rosset et al. [2010] found patient IDs to provide strong predictive cues about a patient's likelihood to have cancer. They incorporated this as a feature to train their model demonstrating improved performance. Rodríguez-Algarra et al. [2019] demonstrate how top performing music labeling systems were exploiting characteristics from the music signal that could not even be heard. In [Charalambous and Bharath, 2016], the accuracy of a gait recognition system was found to drop when confounding factors were removed during training. Mendelson et al. [2017] describes how biases introduced as a result of dataset selection influenced the performance of an Alzheimer's disease classification system. Stowell et al. [2019] studies reducing the effects of confounders in the dataset that bias performance to build robust automatic acoustic individual

[3] "the inclination or prejudice of a decision made by an Artificial Intelligence system which is for or against one person or group, especially in a way considered to be unfair" [Ntoutsi et al., 2020]

identification systems. Even in computer vision applications, [Tommasi et al., 2015] have analysed the effect of biases across several datasets. Furthermore, [Kaufman et al., 2011] describes that data leakage can often lead to overestimation of model performance, producing too-good-to-be-true results. One relevant work in anti-spoofing in this regard is that of Tom et al. [2018] who reported 0% EER on both the development and evaluation sets of the ASVspoof 2017 v1.0 dataset.

Their trustworthiness is therefore called into question and some can behave much like a "horse" in machine learning [Hernandez-Orallo, 2019, Sturm, 2014], i.e. a model that provides excellent results using cues not relevant to the actual problem [Sturm, 2016, Rodríguez-Algarra et al., 2019]. As highlighted in [Rosset et al., 2010], such biases can occur as a result of data collection, compilation, aggregation and partition. Such biases can have a severe impact on the trustworthiness of ML applications, and for domains such as finance, medicine and security (including ASV anti-spoofing) this can be catastrophic. Therefore, it is beneficial to perform an in-depth dataset analysis [Torralba and Efros, 2011, Chen and Asch, 2017], detect the presence of artefacts or confounders [Stowell et al., 2019], ensuring models do not exploit irrelevant factors during training, and therefore yield reliable performance estimates.

In this thesis, a study on dataset artefacts and how they impact ML countermeasure decisions has been carried out in Subsection 4.2.3 (on version 1.0 of the ASVspoof 2017 dataset), Section 4.6 (on version 2.0 of the ASVspoof 2017 dataset) and Subsection 5.2.5 (on the ASVspoof 2019 PA dataset). The details on the two spoofing datasets are provided in Chapter 3.

2.9.2 Understanding model predictions

The ability to provide explanations for model predictions is a key factor towards fairness and accountability of any machine learning model. There exist several methods to understand the global or local behaviour of ML models [Montavon et al., 2018, Mishra, 2020]. While global methods aim at understanding what information a neuron or a group of neurons in any hidden layer has learned during model training, local methods on the other hand aim at generating explanations by highlighting input features that contributed the most for model predictions.

Activation maximization [Erhan et al., 2009] and feature inversion [Mahendran and Vedaldi, 2015] are two popular interpretable machine learning (IML) methods focussed on global analysis of pre-trained ML models. Activation maximization aims at generating synthetic input examples that maximizes the activation of desired neurons in a deep neural network (DNN) [Erhan et al., 2009].

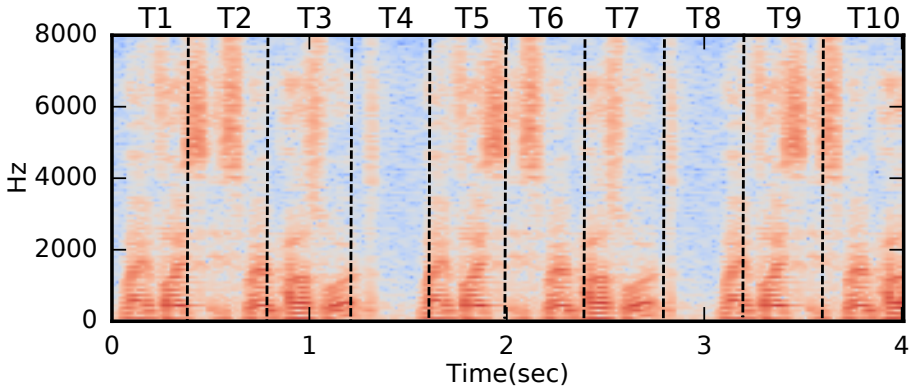


Figure 2.10: Temporal segmentation of an input spectrogram ($\mathbf{x}_i$) into 10 uniform segments (Ti), each of duration 400 ms.

Feature inversion on the other hand aims at learning a mapping function to transform learned features (for example deep features — output from any hidden layer) back to the original input space. Learning such mapping helps to understand the information preserved by various layers which in turn helps to understand important features used by the model to form predictions [Mahendran and Vedaldi, 2015, Mishra, 2020].

There exist many methods for analysing ML models locally, for example sensitivity analysis and function decomposition [Mishra, 2020]. We describe one IML method called Sound LIME (SLIME) [Mishra et al., 2017] that is used in this thesis, and is based on sensitivity analysis. SLIME is an algorithm to analyse the local behaviour of any (deep or shallow) machine listening model. SLIME is based on the LIME algorithm [Ribeiro et al., 2016], which refers to Local Interpretable Model-Agnostic Explanations. Ribeiro et al. [2016] introduced the LIME algorithm and demonstrated its applicability to image recognition and text classification models.

SLIME extends LIME to machine listening systems by defining an *interpretable sequence* $\mathcal{X}_i$ for an input instance $\mathbf{x}_i$ (e.g., a time-frequency representation). An interpretable sequence is composed of elements, called interpretable components, that are in some way related to the classification of $\mathbf{x}_i$. SLIME defines three types of interpretable sequences (temporal, spectral, and time-frequency) depending on the way it segments $\mathbf{x}_i$ into interpretable components. For example, a temporal sequence $\mathcal{X}_i^t$ consists of temporal segments that SLIME generates by segmenting $\mathbf{x}_i$ (uniformly or non-uniformly) along the temporal dimension as shown in Fig. 2.10. SLIME maps an input instance $\mathbf{x}_i$ to its interpretable representation $\mathbf{x}_i^* \in \{0,1\}^{|\mathcal{X}_i|}$. In order to generate a local explanation for the prediction $f(\mathbf{x}_i)$ where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a classifier, SLIME first gener-

ates N artificial samples ($\mathbf{z}_i^*$) by perturbing the interpretable representation. SLIME perturbs $\mathbf{x}_i^*$ by randomly setting the interpretable components to zero. For example, for the instance in Fig. 2.10, if we set the temporal segments $T1$, $T4$ and $T7$ to zero, then a possible $\mathbf{z}_i^*$ is given as (0, 1, 1, 0, 1, 1, 0, 1, 1, 1). Later, SLIME maps each perturbed representation $\mathbf{z}_i^*$ to the feature space with an assumption that such a mapping exists. In other words, SLIME assumes that for each $\mathbf{z}_i^*$ there exists a corresponding $\mathbf{z}_i$ in the feature space. Finally, SLIME uses the perturbed representations $\mathbf{z}_i^*$ and their corresponding predictions $f(\mathbf{z}_i)$ to approximate f with a linear model g in the interpretable space $\tau = \{0,1\}^{|\mathcal{X}_i|}$. The explanation to the prediction $f(\mathbf{x}_i)$ is given by the weights $\mathbf{w}$ of the linear model $g(\mathbf{z}^*) = \mathbf{w}^T\mathbf{z}^*; \mathbf{z}^* \in \tau$. Formally, SLIME generates an explanation by the optimisation

$$\min_{g \in G} L(f, g, \rho_{\mathbf{x}_i}) + \Delta(g) \tag{2.14}$$

where L is a loss function (squared error between the original prediction $f(\mathbf{z}_i)$ and the model approximation $g(\mathbf{z}_i^*)$), $\rho_{\mathbf{x}_i}$ measures the distance between the input instance $\mathbf{x}_i$ and the generated sample $\mathbf{z}_i$, and $\Delta(g)$ measures the complexity of g (e.g., sparsity).

2.10 Discussion

This chapter provided a detailed survey on necessary background on spoofing attacks in voice biometrics which is driven by an ASV technology. For this, the chapter provided a brief background on ASV as a starting point of discussion. Although the focus of this thesis is on replay attacks, we briefly described other three forms of spoofing attacks (mimicry, speech synthesis and voice conversion) that have been studied in the literature in Section 2.2. The work in this thesis uses datasets that are released as part of the ASVspoof series — an ASV-community driven automatic speaker verification spoofing and countermeasures challenge focussed on promoting anti-spoofing research for secure ASV. To that end, Section 2.3 described an overview of the ASVspoof series.

After this, the chapter provided a detailed survey on published works towards replay spoofing attacks using the ASVspoof datasets and the ones before the ASVspoof series began in 2015. The description has been separated into two groups for better readability. Subsection 2.4.1 summarised traditional methods which used classical signal processing methods for feature extraction and shallow classifiers for modelling. In contrast, Subsection 2.4.2 provided background on deep learning based countermeasures for replay attack detection. To be precise, all works that used deep learning either for feature learning, or as a classifier, or for end-to-end modelling were summarised in this subsection. Following this,

Sections 2.5, 2.6, and 2.7 provided a detailed background of different signal processing methods, discriminative backend classifiers and generative backend classifiers that are used in this thesis for replay spoofing detection. A survey on subband modelling applied to audio and speech applications including ASV anti-spoofing was provided in Section 2.8.

Finally, this chapter also provided a survey on trustworthy machine learning and discussed the importance of trustworthy countermeasures for spoofing detection (Section 2.9). A summary on how dataset artefacts and confounders influence decisions of a machine learning model was provided in Subsection 2.9.1. An overview of SLIME, a method from the interpretable machine learning literature was provided in Subsection 2.9.2 and will be used later in Section 4.5.

Chapter 3

Spoofing corpus and evaluation metrics

3.1 Introduction

This chapter describes the corpus and the evaluation metrics used to study replay spoofing countermeasures. The thesis mostly uses the publicly available datasets released by the ASV community as part of the ongoing bi-annual ASVspoof challenge (as described in Section 2.3). Firstly, Section 3.2 describes the ASVspoof 2017 dataset. This is the first publicly available replay spoofing dataset designed by playing back bonafide audio utterances and re-recording them in real 'wild' acoustic conditions. It will be extensively used in both Chapter 4 and Chapter 5 for the analysis and design of replay spoofing countermeasures. Both version 1.0 and version 2.0 of the ASVspoof 2017 dataset are described in Subsections 3.2.1 and 3.2.2, respectively. Furthermore, Subsection 3.2.3 provides results of qualitative analysis performed on version 2.0 of the dataset bringing interesting insights that might help understanding a countermeasure trained on this dataset. Then Section 3.3 describes the ASVspoof 2019 dataset which was released after the 2019 challenge evaluation. This evaluation focused on two sub-tasks: logical access (LA) and physical access (PA) spoofing attack conditions. These sub-tasks along with the LA and PA datasets are discussed in Subsections 3.3.1 and 3.3.2 respectively. Furthermore, a brief description of the real ASVspoof PA test set is provided in Subsection 3.3.3. Then Section 3.4 provides a brief discussion on other publicly available spoofing datasets. The next section (Section 3.5) describes different metrics used in this thesis for evaluation of countermeasure model performance. These metrics are adopted from the evaluation metrics used in the ASVspoof challenges. Finally,

Table 3.1: Phrases used in the ASVspoof 2017 dataset.

Phrase Id	Description
S01	My voice is my password
S02	Ok google
S03	Only lawyers love millionaires
S04	Artificial intelligence is for real
S05	Birthday parties have cupcakes and ice cream
S06	Actions speak louder than words
S07	There is no such thing as a free lunch
S08	A watched pot never boils
S09	Jealousy has twenty-twenty vision
S10	Necessity is the mother of invention

this chapter concludes with a summary in Section 3.6.

3.2 ASVspoof 2017 dataset

3.2.1 Version 1.0

The ASVspoof 2017 challenge evaluation[1] used version 1.0 of the dataset with the same name. It is derived from *RedDots* [Lee et al., 2015] which is a text dependent speaker verification dataset. Part 01 of the RedDots corpus comprised 10 common short phrases, as detailed in Table 3.1, were used to simulate replay attacks. Several of these short utterances were first concatenated using a segment marker to obtain a long utterance, which were then played back and recorded through different types of playback and recording devices [Kinnunen et al., 2017b]. Dual-tone multi-frequency (DTMF) sound was used as a segment boundary marker. Such playback and re-recording was performed in diverse acoustic conditions to capture realistic replay attack conditions. The DTMF marker was later used to retrieve individual replayed utterances. The database is divided into three subsets: training, development and evaluation and is summarised in Table 3.2. The speakers in the three subsets are non-overlapping and there are only male speakers in this corpus. Meta-data such as class labels, speaker ID, phrase ID and replay configurations[2] are available only for the training and development sets. A total of 15 playback devices (P01-P15) and 16 recording devices (R01-R016) were used to develop v1.0 of the corpus. Acoustic environment details used to simulate such attacks is not publicly available. Furthermore, only the class labels and phrase IDs are available for the evaluation set. See [Kinnunen et al., 2017a] for a summary of the ASVspoof

[1] https://www.asvspoof.org/index2017.html

[2] Replay configuration: a unique combination of recording device (R), playback device (P) and acoustic environment (E).

Table 3.2: The ASVspoof 2017 v1.0 dataset statistics. * in hours. RC: replay configuration.

Subset	#Speakers	#RC	# Bonafide	#Spoofed	Duration*
Train	10	3	1508	1508	2.22
Dev	8	10	760	950	1.44
Eval	24	110	1298	12922	11.95

Table 3.3: Same as in Table 3.2 but for the ASVspoof 2017 v2.0 dataset.

Subset	#Speakers	#RC	# Bonafide	#Spoofed	Duration*
Train	10	3	1507	1507	2.22
Dev	8	10	760	950	1.44
Eval	24	57	1298	12008	11.94

Table 3.4: Acoustic environments used in the ASVspoof 2017 v2.0 dataset.

ID	Environment	Quality	ID	Environment	Quality
E01	Anechoic room	High	E14	Office 02	Medium
E02	Balcony 01	Low	E15	Office 03	Medium
E03	Balcony 02	Low	E16	Office 04	Medium
E04	Home 07	Medium	E17	Office 05	Medium
E05	Home 08	Medium	E18	Office 06	Medium
E06	Canteen	Low	E19	Office 07	Medium
E07	Home 01	Medium	E20	Office 08	Medium
E08	Home 02	Medium	E21	Office 09	Medium
E09	Home 03	Medium	E22	Office 10	Medium
E10	Home 04	Medium	E23	Studio	High
E11	Home 05	Medium	E24	Analog wire 01	High
E12	Home 06	Medium	E25	Analog wire 02	High
E13	Office 01	Medium	E26	Analog wire 03	High

2017 evaluation and [Kinnunen et al., 2017b] for further details on v1.0 of this dataset.

3.2.2 Version 2.0

Post-evaluation, as described in Subsection 4.2.3, some issues with v1.0 of the dataset were identified that biased model performance. It was demonstrated that the knowledge of a simple class-dependent cue — silence frames of zeros present in some of the bonafide files but missing in spoofed ones, in the v1.0 dataset can easily compromise class decisions. These findings were reported to the challenge organisers (see Section 4.2.3), and an updated version 2.0 [Delgado et al., 2018] dataset fixing these anomalies was subsequently released by the organisers. Table 3.3 summarises the v2.0 dataset statistics. Two training audio files $T\_1001658.wav$ and $T\_1000150.wav$ that do not contain any speech

Table 3.5: Playback devices used in the ASVspoof 2017 v2.0 dataset.

ID	Playback devices	Quality
P01	All-in-one PC speakers	Medium
P02	Creative A60 speakers	Medium
P03	Genelec 8020C studio monitor	High
P04	Genelec 8020C studio monitor (2 speakers)	High
P05	Beyerdynamic DT 770 PRO headphones	High
P06	Dell laptop internal speakers	Low
P07	Dynaudio BM5A speaker	High
P08	HP Laptop internal speakers	Low
P09	VIFA M10MD-39-08 speaker	High
P10	ACER netbook internal speakers	Low
P11	BQ Aquaris M5 smartphone	Low
P12	Logitech low quality speakers	Medium
P13	Desktop PC line output	High
P14	Labtec LCS-1050 speakers	Medium
P15	Edirol MA-15D studio monitor	High
P16	Lenovo Ideatab S6000-H tablet	Low
P17	Logitech S120 multimedia speakers	Low
P18	MacBook pro internal speakers	Low
P19	Altec lansing Orbit USB iML227 portable speaker	Medium
P20	Samsung GT-I9100 smartphone	Low
P21	Samsung GT-P6200 tablet	Low
P22	Behringer Truth B2030A studio monitor	High
P23	Focusrite Scarlett 2i2 audio interface line output	High
P24	Focusrite Scarlett 2i4 audio interface line output	High
P25	Genelec 6010A studio monitor	High
P26	AKG K242HD Headset	High

were removed. Also, 914 corrupted spoofed recordings were removed from the evaluation set in the v2.0 dataset as can be seen from the table. Furthermore, a number of changes were applied in terms of replay configurations (RCs). A total of 26 different acoustic environments (E01 - E26), 26 playback devices (P01 - P26) and 25 recording devices (R01 - R25) were used to compile v2.0 of this dataset. However, only 61 unique RCs are used in v2.0 of the dataset from a space of $26 \times 26 \times 25$ possible combinations after grouping together the overlapping configurations.

As presented in Table 3.3, the training set has three RCs. There are no overlapping RCs between the training and development sets. However, one RC (E21 P03 R01) from the evaluation set is present in the training set. The development set has ten RCs out of which seven RCs (E16 P07 R06, E16 P07 R05, E16 P07 R07, E06 P09 R06, E06 P09 R05, E06 P09 R07, E18 P05 R03) overlap with the ones in the evaluation set. It is worth noting that though there are some overlaps in terms of RCs, however, the speakers are disjoint across different

Table 3.6: Recording devices used in the ASVspoof 2017 v2.0 dataset.

ID	Recording devices	Quality
R01	Zoom H6 handy recorder	High
R02	BQ Aquaris M5 smartphone	Low
R03	Low-quality headset	Medium
R04	Nokia Lumia 635 smartphone	Low
R05	Røde NT2 microphone	High
R06	Røde smartLav+ microphone	High
R07	Samsung Galaxy S7 smartphone	Low
R08	Desktop PC microphone input	High
R09	Zoom H6 recorder with Behringer ECM8000 mic.	High
R10	Zoom H6 recorder with MSH-6 microphone	High
R11	Zoom H6 recorder. with XY microphone	High
R12	iPhone 5c smartphone	Low
R13	iPhone 7 plus smartphone	Low
R14	iPhone 4 smartphone	Low
R15	Logitech C920 webcam	Medium
R16	miniDSP UMIK-1 microphone	High
R17	Samsung Galaxy Trend 2 smartphone	Low
R18	Samsung GT-I9100 smartphone	Low
R19	Samsung GT-P6200 tablet	Low
R20	Samsung Trend 2 smartphone	Low
R21	AKG C3000 microphone	High
R22	SE electronic 2200a microphone	High
R23	Focusrite Scarlett 2i2 interface line input	High
R24	Focusrite Scarlett 2i4 interface line input	High
R25	Zoom HD1 handy recorder	High

sets. Delgado et al. [2018] grouped the replay configurations of the evaluation set into three categories depending on the level of threat they present to an ASV system. (1) **Low**, signifies the use of a low quality RD, PD and noisy AE (eg. balcony) to simulate a replay attack. (2) **Medium**, signifies the use of a medium quality RD, PD and a medium noise AE (eg. office). (3) **High**, indicates the use of a high quality RD, PD and a low noise AE (eg. studio). Low quality replay recordings (with high background noise, reverberation) are assumed to pose the least threat to ASV systems in contrast to high quality replay recordings. Tables 3.4, 3.5 and 3.6 summarises the environment, recording devices and playback devices used to simulate replay attacks and create the ASVspoof 2017 dataset. For further details please refer to [Delgado et al., 2018].

3.2.3 Qualitative analysis of v2.0

It was found that the issues on the ASVspoof 2017 dataset were not addressed completely in the updated version. As highlighted in Section 4.5, v2.0 may still have some issues. Therefore, this section performs qualitative analysis on

all the audio recordings in the training and development sets of version 2.0 of the dataset. As for the evaluation set, this analysis was only performed on the bonafide recordings. Due to the large number of spoof recordings (about $13,000$) in the evaluation set, manual inspection on them was not possible. These findings are categorised into two classes: unexpected/unnatural and expected/natural, and are described next.

Expected/Natural

Though some of the observations described here are natural for a dataset, these insights might help in understanding model behaviours.

- *Mispronunciation and/or incomplete sentence.* Mispronounced words and incomplete sentences are found in some audio recordings. The words are missing either in the beginning or at the end of an utterance. The training and development sets have 11 and 98 such audio files and 2 files in the evaluation set (bonafide class).

- *Unwanted noise/speech.* Different from a *burst click sound* (BCS[3]), audio recordings containing short duration noise or speech (different from 10 phrases used in the corpus) in the start and/or at the end of an utterance is found. The training set has 11 such files (4 bonafide and 7 spoof), the development set has 70 (28 bonafide and 42 spoof), and 8 bonafide files are found in the evaluation set.

- *Sentence S02 - "Ok Google".* It is one of the phrases used in the ASVspoof 2017 dataset with an average duration between 0.7 - 0.8 seconds. We find 165, 136 and 1282 audio examples of S02 in the training, development and evaluation sets with more than 1.5 seconds duration. This suggests that more than half of the contents of each recording contain noise or nonspeech.

Unexpected/Unnatural

The observations described here as unnatural or unexpected for a dataset could be due to errors/faults made during data collection and compilation. As demonstrated later in Section 4.6.3, some of them have a profound impact on model decisions raising concerns on the validity of results reported in the literature [Lavrentyeva et al., 2017, Tom et al., 2018, M S and Murthy, 2018, Suthokumar et al., 2018].

[3]BCS is defined as an abrupt click sound.

- *Pattern difference.* This thesis uses the term pattern difference (applied to the ASVspoof 2017 v2.0 dataset) as the *presence or absence of nonspeech*[4] *in the first* 300 *milliseconds* between bonafide and spoof recordings. As Section 4.6.3 demonstrates, this pattern difference has a profound impact on model decisions. About 60.45%, 73.55% and 69.1% of the bonafide audio files in the training, development and evaluation[5] sets respectively have nonspeech. On the contrary, 68.74% and 41.05% of the spoof files in the training and development sets respectively have speech occurring within the first 300 ms.

- *Burst click sound (BCS).* This thesis uses the term BCS to define an abrupt click sound (low or loud) found in the start of audio recordings. About 36.36%, 23.55% and 41.06% of bonafide audio files in the training, development and evaluation sets were found to contain BCS in the start. On the contrary, 2.45% spoof files in the training set have BCS. No spoof class audio files in the development set have such BCS, and we do not have ground truth annotations for spoofed signals in the evaluation set.

- *Dual-tone multi-frequency signaling (DTMF) sound.* About 45.58% (687 out of 1507) of spoof audio files in the training set and 16.63% (158 out of 950) in the development set were found to contain a DTMF sound (low or loud) within the first 200-250 ms. The DTMF sound often overlaps with the actual spoken speech. We find 33.77% (232 out of 687) spoof files and 6.96% (11 out of 158) in the training and development sets contain such overlapping sounds. On the contrary, the bonafide class audio files do not have such DTMF sounds.

- *Silence.* We find some bonafide audio recordings with more than 10 ms zero valued silence in their start. There are 19.11% (288 out of 1507) such bonafide files in the training set, 1.97% (15 out of 760) in the development set and 10.09% (131 of 1298) in the evaluation set. Furthermore, in the training set we find 23.61% (68 out of 288) files have more than 70 ms silence and 12.85% (37 out of 288) with more than 100 ms silence in the start. In contrast no spoof class files are found to have such zero valued silence.

- *Corrupted audio files.* Bonafide files T_1000788.wav and D_1000581.wav, and spoof files T_1002296.wav and E_1011601.wav do not contain any speech. The prefixes T, D and E used in audio files denote training, development and evaluation sets respectively.

[4]We use the *nonspeech* term to imply noise, music or silence samples in the start.

[5]Since we could not inspect a large number of spoof test files we do not have pattern difference statistics on them.

Table 3.7: ASVspoof 2019 LA dataset statistics.

Subset	#Speakers	#Bonafide	#Spoofed
Training	20	2580	22800
Development	20	2548	22296
Evaluation	67	7355	63882

The filelists corresponding to above findings are provided in `https://zenodo.org/record/3601188#.XmzuknX7TCI`.

3.3 ASVspoof 2019

The ASVspoof 2019 dataset was released as part of the third ASVspoof challenge evaluation held in 2019. It comprises two subtasks: Logical access (LA) and physical access (PA). The LA subtask involves spoofing attacks mounted by injecting synthetic/converted speech directly into an ASV system pipeline bypassing its microphone. The PA subtask on the other hand involves physical transmission of impersonated or playback speech through the systems' microphone. Examples of LA attacks include TTS and VC. Replay and mimicry are examples of PA attacks. However, the ASVspoof 2019 PA subtask includes only replay attacks. Two sub-datasets for the LA and PA conditions were made available publicly[6] by the ASVspoof organisers. As in previous ASVspoof challenge editions, the main task in this challenge is to build a standalone spoofing detection system. However, with the introduction of the tandem detection cost function (t-DCF) [Kinnunen et al., 2018] metric explained in Section 3.5.2, ASV system scores were provided to every participants making joint evaluation of ASV and spoofing countermeasures possible. Therefore, unlike previous challenges, this challenge made use of t-DCF as a primary evaluation metric and equal error rate (EER) as the secondary metric. Section 3.5 provides further details on evaluation metrics.

The ASVspoof 2019 database for LA and PA is derived from a standard multi-speaker speech synthesis database called voice cloning toolkit database[7]. A total of 46 male and 61 female speakers were used to collect clean bonafide speech recordings without any background noise or channel effects. Spoofed speech is then derived from them by applying advanced state-of-the-art TTS, VC algorithms (for the LA dataset) and simulating replay attacks under controlled simulation conditions (for the PA dataset).

[6]https://datashare.is.ed.ac.uk/handle/10283/3336
[7]http://dx.doi.org/10.7488/ds/1994

Table 3.8: ASVspoof 2019 PA dataset statistics.

Subset	#Speakers	#Bonafide	#Spoofed
Training	20	5400	48600
Development	20	5400	24300
Evaluation	67	18090	116640

3.3.1 Logical access (LA) spoofing dataset

Here, LA attack simply refers to TTS and VC attacks. The main difference between the 2019 and 2015 editions' TTS and VC datasets is in the use of algorithms for creating them. The ASVspoof 2015 dataset was created using state-of-the-art TTS and VC algorithms available until 2015. However, since 2015, the TTS and VC communities both have made a substantial progress showcasing models capable of producing synthetic and converted speech that are virtually indistinguishable from real human speech. The ASVspoof 2019 LA dataset is therefore created to bridge this gap and study TTS and VC based spoofing algorithms using advanced state-of-the-art methodologies developed since 2015 [ASVspoof 2019 evaluation_plan]. Table 3.7 summarises the ASVspoof 2019 LA dataset. Both the training and development sets consist of 8 male and 12 female speakers [Todisco et al., 2019., ASVspoof 2019 evaluation_plan].

Furthermore, it should be noted that the main focus of this thesis is on replay attacks, and therefore, most of the work is based on the ASVspoof 2017 and 2019 PA dataset. The LA dataset was used only during our participation in the ASVspoof 2019 challenge evaluation (Section 5.2).

3.3.2 Physical access (PA) spoofing dataset

Understanding the replay spoofing attack problem using the ASVspoof 2017 edition dataset was quite challenging due to the methodology adopted for data collection and compilation [ASVspoof 2019 evaluation_plan]. It was created from real playback and re-recording of bonafide utterances from *RedDots* [Lee et al., 2015] in a somewhat uncontrolled setup [Todisco et al., 2019., ASVspoof 2019 evaluation_plan]. To overcome these issues and perform better analysis towards understanding the replay spoofing problem, the ASVspoof 2019 PA dataset was created using simulations in a much more controlled setup. Replay attacks were simulated using a variety of replay and recording devices under carefully controlled room acoustic and replay configurations as detailed in [ASVspoof 2019 evaluation_plan]. Here, PA attack simply refers to replay spoofing attacks. Table 3.8 summarises the PA dataset. As in LA, the training and development sets consist of 8 male and 12 female speakers [Todisco et al., 2019., ASVspoof 2019 evaluation_plan]. However, the numbers of audio

examples in the training and development sets are comparatively larger than in LA.

3.3.3 Real PA dataset

While the ASVspoof 2019 PA dataset was created using *simulated* replay attacks, the ASVspoof 2019 real PA dataset consists of audio recordings developed under real replay conditions. The real PA dataset consists of $2,700$ audio files with 540 bonafide and $2,160$ spoof recordings [Todisco et al., 2019.]. The replayed recordings were developed using: 10 different recording devices and 7 different playback devices of low and high qualities; three different types of acoustic environments: medium size office, large office and a meeting room; four different noise conditions to add additive noise: very quiet, quiet fan, low AC noise, and window open; close and far distances between ASV-talker and attacker-talker[8]. This testset is primarily developed for studying the generalisability of countermeasure models in unseen real-world test conditions.

3.4 Other spoofing corpora

For completeness, this section will now provide a brief description of the other spoofing datasets that are available publicly but not used extensively in this thesis. Firstly, the realistic replay attack corpus (ReMASC), a new replay spoofing dataset, is described in Subsection 3.4.1. Then the Subsection 3.4.2 discusses the AVspoof dataset which is the first publicly available corpus containing replay attack samples created in a controlled recording and playback setting. Another publicly available spoofing dataset is ASVspoof 2015 that was released in 2015 as part of the first ASVspoof competition [Wu et al., 2015c]. This chapter does not provide much details here because this thesis focusses on replay spoofing attacks and the ASVspoof 2015 dataset was designed to study logical access (TTS, VC) spoofing attack conditions. Please see [Wu et al., 2015c] for more details on the ASVspoof 2015 challenge and the dataset.

3.4.1 ReMASC

ReMASC - a realistic replay attack corpus [Gong et al., 2019] is a publicly available corpus for replay spoofing attack research in voice controlled applications. It consists of both bonafide and replayed recordings collected under realistic use-case scenarios. The dataset contains recordings collected using a variety of microphone arrays recorded in different acoustical environmental conditions,

[8]Details taken from "README.PA.read.txt" file released along with the real PA dataset.

with various types of background noise and different positions between speakers and VC systems. More details on this dataset can be found in [Gong et al., 2019]. The current dataset version consists of two disjoint sets. The first set has about $2,000$ replayed recordings covering all recording conditions for quick evaluation (in a cross-dataset setting) of countermeasure models. The second set consists of about $27,000$ recordings which can be used to design countermeasure models and further analyse how different playback devices and microphones impact model performance.

3.4.2 AVspoof

The AVspoof (audio visual spoofing) dataset is the first publicly available corpus that includes replay spoofing attacks [Ergünay et al., 2015]. It contains bonafide speech samples recorded from 44 speakers (31 males and 13 females) using one laptop and two smartphones. For controlled dataset creation the authors made an assumption that an ASV system is deployed in a laptop. The replayed samples were then generated in three different settings. First, the original bonafide samples were replayed using a laptop with internal speakers and external high quality (HQ) speakers, and also with two smartphones (Samsung Galaxy S4 and iPhone 3G). Second, the synthesized speech produced using TTS was replayed using a laptop and HQ speakers to an ASV system (installed in the laptop). Thirdly, voice converted speech was replayed with a laptop with HQ speakers. See [Ergünay et al., 2015] (Table 1) for further details on the distribution of replayed and bonafide samples. It should be noted that both bonafide and replayed utterances were compiled in a controlled setting. This is one reason why this thesis does not use this dataset, rather it focusses on ASVspoof datasets due to its popularity and incorporation of much more wild replay attack conditions. However, for completeness, this thesis includes the details of the AVspoof dataset here.

3.5 Evaluation metrics

So far this chapter has provided an overview of publicly available replay spoofing datasets that will be used in this thesis. Now, to evaluate the performance of countermeasure models it is important to define appropriate evaluation metrics. To this end, the thesis adopts the two metrics used in the ASVspoof challenges: equal error rate (EER) and tandem detection cost function (t-DCF). Section 3.5.1 and Section 3.5.2 provide further details on these metrics.

Unless otherwise stated, the EER is used as a primary evaluation metric in this thesis. Furthermore, the EER and t-DCF metrics are computed using the

scripts released by the organisers of the ASVspoof 2019 challenge.

3.5.1 Equal error rate

The equal error rate (EER) metric is used to evaluate the ability of spoofing countermeasure models to discriminate bonafide and spoofed utterances from each other. EER was the primary evaluation metric of the ASVspoof 2017 challenge, and a secondary metric of the ASVspoof 2019 challenge. EER is the error rate at an operating point where the false acceptance (false alarm) and false rejection (miss) rates are equal. The false acceptance rate (FAR) and the false rejection rate (FRR) are respectively defined as:

$$\text{FAR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.1}$$

$$\text{FRR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{3.2}$$

where TP, TN, FP, and FN denote true positive, true negative, false positive and false negative counts respectively. It should be further noted that a reference value of 50% EER indicates the chance level.

3.5.2 Tandem detection cost function

In addition to EER, this thesis uses the tandem detection cost function (t-DCF) [Kinnunen et al., 2018] metric to evaluate countermeasure (CM) performance. Unlike EER, which evaluates countermeasure performance in isolation from ASV, the t-DCF metric evaluates countermeasure and ASV performance jointly under a Bayesian decision risk approach. Let s and t represent detection thresholds for a CM and an ASV system respectively. The t-DCF metric is defined as:

$$\begin{aligned} \text{t-DCF}(s,t) = {} & C_{\text{miss}}^{\text{asv}} \cdot \pi_{tar} \cdot P_a(s,t) \\ & + C_{\text{fa}}^{\text{asv}} \cdot \pi_{non} \cdot P_b(s,t) \\ & + C_{\text{fa}}^{\text{cm}} \cdot \pi_{spoof} \cdot P_c(s,t) \\ & + C_{\text{miss}}^{\text{cm}} \cdot \pi_{tar} \cdot P_d(s) \end{aligned} \tag{3.3}$$

where, π_{tar}, π_{non}, π_{spoof} represents target, nontarget and spoof prior probabilities; $P_a(s,t)$, $P_b(s,t)$, $P_c(s,t)$ and $P_d(s)$ represent 4 different error probabilities represented as a function of detection thresholds and defined as follows. $P_a(s,t)$: CM does not miss human speech, and ASV falsely rejects the target; $P_b(s,t)$: CM does not miss human speech, and ASV falsely accepts the nontarget; $P_c(s,t)$: CM falsely passes on a spoof sample, and ASV does not miss the target; $P_d(s)$: CM misses human speech; $C_{\text{miss}}^{\text{asv}}$: cost of ASV system rejecting a

target trial[9]; $C_{\text{fa}}^{\text{asv}}$: cost of ASV system accepting a nontarget trial; $C_{\text{fa}}^{\text{cm}}$: cost of CM accepting a spoof trial; $C_{\text{miss}}^{\text{cm}}$: cost of CM rejecting a human trial.

The same t-DCF cost and prior parameters as used in the ASVspoof 2019 evaluation [Todisco et al., 2019., ASVspoof 2019 evaluation_plan], with the x-vector probabilistic linear discriminant analysis (PLDA) scores provided by the organisers of the same challenge is used in this thesis. The ASV system is set to its EER operating point while the (normalized) t-DCF is reported by setting the countermeasure to its minimum-cost operating point. A reference value 1.00 of (normalized) t-DCF indicates an uninformative countermeasure. For further details please see [Kinnunen et al., 2018].

3.6 Summary

This chapter provided a detailed description of the spoofing datasets and evaluation metrics available publicly for promoting anti-spoofing research. The work described in Chapters 4 and 5 mostly uses the ASVspoof 2017 and 2019 datasets. Furthermore, for completeness, the chapter also included a brief description of datasets not used in this thesis but available publicly. Along with the database description, this chapter also summarised the results of a qualitative analysis on ASVspoof 2017 v2.0 that has provided interesting insights into the dataset. These insights will be later used in Section 4.6 to understand how they influence decisions made by countermeasure models. Finally, this chapter provided a description of two different performance metrics that will be used in this thesis to evaluate countermeasure performance.

[9]Trial: is simply a test speech utterance passed to a CM or an ASV system.

Chapter 4

Analysis of spoofing countermeasures

4.1 Introduction

This chapter presents a series of studies on the analysis of existing features, classifiers and methods for replay spoofing detection. It aims at serving as the basis towards understanding the replay spoofing attack problem by first exploring existing methodologies and techniques from the literature. The work reported in this chapter mostly uses the ASVspoof 2017 dataset, as the ASVspoof 2019 dataset was released towards the completion of this thesis. Section 3 describes these datasets.

Firstly, in Section 4.2 several existing hand-crafted features, coupled with a GMM classifier, that showed promising performance on the ASVspoof 2015 dataset for the detection of converted and synthetic speech, are considered. Their generalisability towards replay spoofing attack detection is investigated using the ASVspoof 2017 v1.0 dataset, followed by the analysis of the best performing GMM countermeasure model. This work is a result of our participation in the ASVspoof 2017 challenge, and was published in [Chettri and Sturm, 2018].

Following the impressive performance by Lavrentyeva et al. [2017] in the ASVspoof 2017 challenge, Section 4.3 focusses on replicating their best performing light CNN (LCNN) model. It also investigates alternative network architectures and further studies how performance varies across different network parameterisations. This work was published in [Chettri et al., 2018b]. The next Section 4.4 then analyses different replay attack conditions and their impact on both frame-level and utterance-level countermeasure models. This work was published in [Chettri et al., 2018c]. Subsequently, Section 4.5 develops a CNN

countermeasure model whose architecture and input representation is adapted from the LCNN model of the ASVspoof 2017 challenge. The SLIME algorithm (described in Section 2.9.2) is then applied to understand what the CNN model has learned to make spoofing decisions. This work was published in [Chettri et al., 2018a].

Following the findings about potential dataset artefacts on the ASVspoof 2017 v2.0 dataset presented in Section 3.2.3, the next Section 4.6 performs an in-depth study through various interventions towards understanding the impact of different dataset artefacts (see Section 3.2.3) on both frame-level and utterance-level countermeasure models. Finally, this chapter concludes with a summary in Section 4.7.

4.2 Generalisability of hand-crafted features

4.2.1 Introduction

This section describes the work towards studying the effectiveness of six different hand-designed features and machine learning approaches for the automatic detection of replayed speech on the ASVspoof 2017 v1.0 dataset. These features have shown good performance on the detection of synthetic and voice converted spoofed speech applied to the ASVspoof 2015 dataset. Replay attacks, however, are in principle very different from these artificial speech attacks: while they have the same objective — to bypass an ASV system — they are acoustically different. Therefore, it is not obvious whether one should expect the extensive prior results reported in [Sahidullah et al., 2015] on the ASVspoof 2015 data to generalise at all to the detection of replay attacks. Thus, our primary scientific contribution in this section is to thoroughly assess performance of these features for replay spoofing detection using the ASVspoof 2017 v1.0 dataset. The work described here is a result of our participation in the ASVspoof 2017 evaluations (described in Section 2.3), and our post-evaluation findings which were published in [Chettri and Sturm, 2018]. Motivated from [Sahidullah et al., 2015], this section focusses on using a Gaussian Mixture Model (GMM) as a backend classifier. Subsection 4.2.2 provides a brief description of six different features investigated and the backend GMM classifier. The performance of these systems on the replay attack problem applied to the ASVspoof 2017 v1.0 dataset is then discussed. The next Subsection 4.2.3 then aims at understanding the best performing GMM countermeasure model and brings interesting findings to light. It is demonstrated that GMM performance can depend on a simple class-dependent cue in the dataset: initial silence frames of zeros appear in the bonafide signals but are missing in the spoofed versions. Furthermore, it is

shown that using this cue, model predictions can be easily fooled. Finally, this section investigates whether this problem can be mitigated by a simple preprocessing on the audio signals. Subsection 4.2.4 then provides a summary of the work done in this section.

4.2.2 Experimental design and evaluation

This section provides a brief description of different features considered, the backend GMM classifier, and the dataset and performance metrics used for performance evaluation. A brief description of the ASVspoof 2017 challenge baselines is also provided.

Feature extraction

We explore the use of six different hand-crafted features: MFCCs, IMFCCs, RFCCs, LFCCs, SCMCs and CQCCs. While the first five features are based on the short-time Fourier transform (STFT), CQCCs use the constant-Q transform. Section 2.5 provides more details on these features. As for feature extraction, we use the publicly available scripts provided by the ASVspoof 2017 challenge organisers for computing CQCC features [Todisco et al., 2017]. For computing the other five STFT-based features, we use the publicly released scripts by Sahidullah et al. [2015]. All our systems use 40-dimensional features obtained by concatenating 20 delta and 20 acceleration coefficients, including energy[1]. We do not use voice activity detection or normalisation.

Model

We use a Gaussian mixture model (GMM) backend to evaluate the generalisability of hand-crafted features for replay spoofing attack detection. Given a speech utterance s, the main goal is to build a system that determines if it is bonafide speech or a replayed recording. Each system except the CQCC[2], one extracts a series of Hamming-windowed frames of 20 ms duration with 50% overlap, and transforms it into a series of T feature vectors, $\mathcal{X}(s) := (\mathbf{x}_1, \ldots, \mathbf{x}_T)$. The system then computes a mean log-likelihood score by

$$\Lambda(s) := \frac{1}{|\mathcal{X}(s)|} \sum_{\mathbf{x}_t \in \mathcal{X}(s)} \log \frac{p(\mathbf{x}_t|\theta_{bonafide})}{p(\mathbf{x}_t|\theta_{spoof})} \tag{4.1}$$

[1] Our choice of feature configuration was optimised on the development set, and we do not use static features.

[2] CQCC features are extracted using the same feature parameterisation as in Todisco et al. [2017].

Table 4.1: GMM performance (EER%) on the ASVspoof 2017 v1.0 development and evaluation sets. B_1: baseline. na: not available.

	IMFCC	MFCC	LFCC	RFCC	SCMC	CQCC	B_1
Dev	8.5	7.17	3.33	5.15	5.46	1.51	*na*
Eval	17.43	26.02	17.61	16.67	14.82	17.78	24.77

where $p(\mathbf{x}|\theta_{bonafide})$ is the probability density characterizing genuine speech features, and $p(\mathbf{x}|\theta_{spoof})$ is that of spoofed speech features. The larger $\Lambda(s)$ is, the more confidence the model has that s is genuine. We estimate $p(\mathbf{x}|\theta_{bonafide})$ and $p(\mathbf{x}|\theta_{spoof})$ by a GMM using the expectation maximization algorithm [Bishop, 2006] on pooled training data. We optimise the number of mixture components on the development dataset and chose 512 for MFCC, LFCC and CQCC; 128 for IMFCC and RFCC; and 256 for SCMC features.

Dataset and evaluation metric

This study uses the ASVspoof 2017 v1.0 dataset (described in Section 3.2.1) that was released as part of the second ASVspoof challenge evaluation (Section 2.3) and the EER metric for performance evaluation as described in Section 3.5.1.

Baseline

Two baseline GMM systems based on CQCC features were provided by the ASVspoof 2017 challenge organisers. Both use 90-dimensional features obtained by combining the 0^{th} and 29 static coefficients, their delta and acceleration coefficients. The first baseline (B_1) uses both training and development data for GMM training while the second baseline (B_2) uses only the training data. However both GMMs used 512 mixture components. Since we use pooled data in this work, we only include B_1 in our further discussion.

Results

Table 4.1 shows the results of our six frame-based GMM systems on both the development and evaluation sets, and the baseline B_1 system. Our CQCC feature based GMM outperforms the baseline performance by a large margin. This suggests that static features may not offer substantial discriminative information useful for replay attack detection in contrast to the use of dynamic features. Except for the one using MFCCs, all systems outperform the 24.77% baseline performance on the evaluation data [Kinnunen et al., 2017a] by a large margin. IMFCC features give more emphasis on high frequency information than MFCCs, and seem to have more discriminability. LFCC and RFCC systems equally emphasise all frequency bands and have similar performance. Both

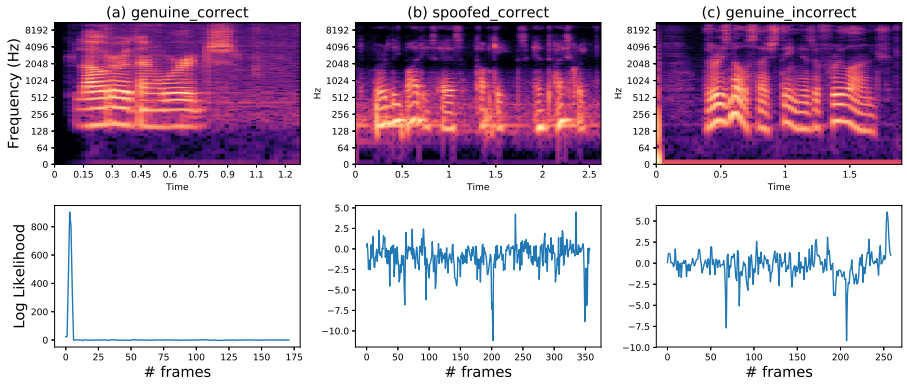


Figure 4.1: Spectrograms (first row) and frame-wise log likelihood score difference (second row) between bonafide and spoof GMMs for (a) genuine_correct; (b) spoofed_correct, and (c) genuine_incorrect audio examples taken from the development set.

CQCC and SCMC features show good generalisability on the replayed speech detection task, but the latter show the best result on the evaluation data. This suggests that the distribution of energy expressed by SCMC features is the most discriminative and generalisable of these six kinds of features.

4.2.3 Analysis

We now take a closer look at the best GMM system which is based on SCMC features to discover the cues that influence its prediction. We look at how the log-likelihood scores for the bonafide (genuine) and spoofed (replayed) GMM models are distributed across frames. We pick a bonafide and spoofed example from the development set that the system confidently and correctly classifies: "*D_1000601.wav*" produces $\Lambda(s) = 14.66$; "*D_1001012.wav*" produces $\Lambda(s) = -0.96$. We also select the genuine signal "*D_1000300.wav*" that is confidently misclassified with a score $\Lambda(s) = -0.21$. For easy reference we define these signals as *genuine_correct*, *spoof_correct* and *genuine_incorrect*. Figure 4.1 shows for each signal its spectrogram (first row) and the frame-wise distribution of log-likelihoods (bottom row) in each model. We observe a marginal difference between genuine and spoofed model scores across frames for *genuine_incorrect* and *spoof_correct*, respectively. However, we see significantly different behavior for *genuine_correct*. The decision for this signal is dominated by its first few frames. We find that many genuine audio files in this dataset contain initial silence frames with zeros which do not appear in the spoofed version. As can be seen in Figure 4.1 second row, the spoofed model assigns a very small probability to such a frame, thus pushing the decision toward the bonafide class. As a

Table 4.2: Performance (EER%) of GMMs after adding the genuine signature to every utterance in the development and evaluation set.

	IMFCC	MFCC	LFCC	RFCC	SCMC	CQCC
Dev	34.54	33.48	34.92	28.92	46.74	2.27
Eval	34.46	35.95	38.23	34.22	44.44	18.71

consequence, this has a large influence on the classifier decision (Equation 4.1).

To further confirm that GMM decisions are influenced by this silence cue favouring the bonafide class, we perform intervention[3] experiments on all frame-based GMM systems. Furthermore, for completeness and to study how this cue affects models trained at utterance-level, we train two such models. First, we train support vector machines (SVMs) using i-vectors which are computed using SCMC features. Second, we train a convolutional neural network (CNN) on power spectrograms using the architecture adapted from the LCNN [Lavrentyeva et al., 2017] model that showed the best performance in the ASVspoof 2017 evaluations.

Frame-based model intervention

We find that *genuine_correct* begins with 60 ms of zeros, except four samples containing non-zero values. Therefore, we define this 60 ms segment of *genuine_correct* as a "genuine signature" and add it onto the beginning of the two other signals, *spoofed_correct* and *genuine_incorrect*. As expected, the model now scores both in favor of being genuine: $\Lambda(s) = 6.85$ and $\Lambda(s) = 11.63$ for *spoofed_correct* and *genuine_incorrect* respectively. When we repeat this process for all test files in the development and evaluation set and re-evaluate all our GMM systems we see a dramatic increase in the EER of all systems except for the CQCC one. The IMFCC system that showed 8.5% and 17.43% EER before gives 34.54% and 34.46% EER on the development and evaluation sets respectively. We observe a similar trend for the LFCC and RFCC systems. Our best performing SCMC system now gives the worst performance. We observe a very small increase in the EER for CQCCs: from 17.78% to 18.81% on the evaluation set in comparison to other five features. Thus, the CQCC features that give higher frequency resolution for lower frequencies and a higher temporal resolution for higher frequencies seem to be robust against such presentation attacks — augmenting silence in the start of test signals.

Our above analysis casts doubts on the reliability of the evaluation results of the ASVspoof Challenge: are the other participating systems benefiting from

[3]This thesis defines ***Intervention*** as a process that updates the original audio signal either by adding or removing audio samples. This is mainly performed to understand the influence of certain dataset related artefacts/biases on model predictions.

Table 4.3: Performance (EER%) for two cases of preprocessing. Approach 1: remove the first 60 ms during testing from all the test files. Approach 2: same as in Approach 1 but retrains the bonafide GMM applying the same preprocessing.

	Approach1		Approach2	
	Dev	Eval	Dev	Eval
IMFCC	8.78	19.18	8.66	19.10
MFCC	8.54	31.79	8.5	31.9
LFCC	4.01	21.46	4.41	21.06
RFCC	7.05	19.85	7.43	20.1
SCMC	6.4	17.98	6.39	17.7
CQCC	2.14	19.79	1.97	19.35

this signature, which will not exist "in the wild"? How prevalent is this signature in the data? Can we improve the reliability of this challenge by simply deleting the first 60 ms of each test audio file, and using the same trained models? To this end, we propose two simple preprocessing approaches. The first approach (Approach 1) involves removing the first 60 ms samples from all test files. The second approach (Approach 2) is similar to the first one but this also involves retraining the bonafide GMM removing the initial 60 ms samples from all the training audio files. Table 4.3 shows the results of this intervention experiment. As can be seen, applying Approach 1 on all the test files increases the EER of each system tested in Table 4.1, but not by a large amount. Furthermore, applying Approach 2, we observe a small increase in the EER of each system. These results suggest that the signature is not very prevalent throughout the data, but that it is prevalent enough to allow a simple means of bypassing an otherwise good performing replay attack spoofing detection system.

Utterance-based model intervention

The previous models we trained and tested on the ASVspoof 2017 v1.0 dataset are all frame-level. Will systems using utterance-level features suffer from the same vulnerability? We now investigate two models: GMM and SVM, built using features learned from a CNN and i-vectors [Dehak et al., 2011]. We do not optimise these models for the best performance.

For the CNN-based features, we use the parameterisation and network architecture from [Lavrentyeva et al., 2017] for training the CNN with the following changes. First, we use a 3 seconds log power spectrogram computed using 2048 FFT points, window size of 128 ms and a 10 ms hop size. We truncate or copy original audio samples to obtain a fixed 3 seconds spectrogram for every audio file in the dataset. Second, we use a convolutional layer in place of a network-in-network layer. Third, we use 64 neurons in the fully connected layer. Fourth,

we replace the max-feature-map by an exponential linear unit [Clevert et al., 2015] activation and train our network. Appendix A provides further details on the model architecture. The trained network extracts 64-dimensional feature vectors for every audio file in the dataset. We then train bonafide and spoofed GMM models using 8 mixture components[4]. Unlike the GMMs trained earlier, this GMM is trained at utterance-level as each audio file is represented by a single 64-dimensional vector. It should be noted that our input pipeline for spectrogram computation uses a preprocessing step that ensures the smallest value in the spectrogram is no less than 1e-7. Thus the network will implicitly take care of the genuine signature (zero-valued samples).

We use 40-dimensional delta-acceleration SCMC features to train a 256 mixture universal background model and total variability matrix with 200 factors on the pooled data. We extract 200-dimensional i-vectors for the entire dataset. We then use the training set i-vectors to train a linear SVM using the Scikit-learn [Pedregosa et al., 2011] library.

Table 4.4: Performance (EER%) of utterance-based models before and after injecting the genuine signature to all the test files in the development and evaluation set. * trained using CNN features. ** trained using i-vectors

	GMM*		SVM**	
	Dev	Eval	Dev	Eval
Before	9.06	32.65	21.88	20.9
After	9.24	32.69	21.81	20.5

Table 4.4 shows the performance of these two utterance-based models before and after we add the "genuine signature" to the test files. As i-vector extraction involves stacking mean vectors from the mixture components, the effect of the zero valued samples is taken care of automatically and thus we do not see any impact on performance after adding the genuine signature. Similarly, the CNN has a max-pooling layer that chooses a maximum from a given block of convolved input, thus the artefacts are taken care of in the first convolutional layer, thereby eliminating the impact of the genuine signature on the predictions. As expected, the experimental results in Table 4.4 clearly indicate that systems trained on utterance-based fixed length feature representations using the ASVspoof 2017 v1.0 dataset are resilient against such presentation attacks.

[4]This choice is motivated from Lavrentyeva et al. [2017] where they used 1-mixture component to train GMM on 32 dimensional CNN features (extracted per utterance). We tried 1, 2, 4 and 8 mixture components and found the best performance using 8 mixture components on the development set.

4.2.4 Discussion

This section investigated the generalisability of six different hand-designed features for the automatic detection of replay spoofing attacks using the ASVspoof 2017 v1.0 dataset. Though these features reported good performance for the detection of synthetic and voice converted speech on the ASVspoof 2015 dataset, they showed poor performance on the ASVspoof 2017 dataset due to the acoustically different problem in hand. Among different features investigated in this section, the SCMC feature based GMMs showed the best replay attack detection performance on the ASVspoof 2017 v1.0 dataset. Deeper analysis (Section 4.2.3) of this system led us to interesting observations. We found the presence of recording artefacts (initial silence frames containing zeros) in some genuine audio files in the dataset that are missing from the replayed version. As a consequence, spoofed models assign a very low likelihood to such frames during testing. We demonstrated how knowledge of such cues can compromise system predictions. Though such data-intrinsic behavior may not appear in real-world scenarios, our work showed the severe impact it can have on the EER for frame-level GMM systems. We investigated two intervention approaches to help mitigate against such manipulation attacks. Comparing Table 4.2 and Table 4.3 we see that our proposed approaches not only helped reduce the error rates of all frame-based GMMs, but they are now more trustworthy. Finally, Section 4.2.3 investigated two utterance-based countermeasure models and shows that they do not suffer from such manipulation. A bigger question we have yet to answer is what is causing the large difference between the EER on the development and evaluation datasets, which we aim to address in the next section.

4.3 CNNs for spoofing detection

4.3.1 Introduction

The previous Section 4.2 studied the generalisability of hand-crafted features that showed good detection performance for spoofed speech produced using text-to-speech and voice conversion techniques, but poor performance for replay spoofing attack detection. This indicates that crafting features incorporating human knowledge might not be easy due to the acoustically different problems, suggesting that data-driven techniques might be more helpful for the replay spoofing detection problem. Many competing systems in the ASVspoof 2017 evaluations used deep learning-based countermeasures for replay spoofing detection. For example, the top two systems [Lavrentyeva et al., 2017] and [Nagarsheth et al., 2017] used deep CNNs as feature extractors to learn discriminative features. Then shallow classifiers such as GMMs and SVMs were trained on them to discriminate between bonafide and spoofed recordings. Among three sub-systems used in score fusion by [Lavrentyeva et al., 2017] as a primary system submission in the 2017 challenge, the best performance was achieved by the light CNN (LCNN) model reporting an EER of 7.34% on the evaluation set of the ASVspoof 2017 v1.0 dataset. The success of CNNs in the ASVspoof 2017 challenge inspires the work reported in this section. Our ultimate goal here is to understand why the LCNN showed remarkable performance towards replay attack detection on this dataset which no other participants could reach a performance even close to 10% EER during the 2017 evaluations. A precursor to analyse such model is to train one that performs 'fairly' (better than the baseline) in the evaluation set. The main focus of this section is therefore on replication of the LCNN model which we aim to use later in Section 4.5 for analysis. Related background on CNNs is provided in Subsection 2.6.2.

To this end, this section reports the experiments and challenges to design a deep countermeasure model that is trained and evaluated on the ASVspoof 2017 v1.0 dataset. Firstly, this section provides a brief motivation of this work with a summary of best deep models published on this dataset. Secondly, Subsection 4.3.2 describes our efforts in replicating the state-of-the-art LCNN model in an end-to-end setting. We found that our CNN-based model generalises well on the development set, but consistently underperforms in the evaluation set. Thirdly we explain our experiments to find a suitable architecture that generalises well to the unseen data in Subsection 4.3.3. We explored a number of architectures including the second best deep model of the challenge [Nagarsheth et al., 2017]. But the performance on the evaluation dataset is always poor. This raises several interesting questions about the possible differences in the dataset

and why they are more evident in an end-to-end setting and what are the possible ways to tackle this problem. We also propose a novel CNN architecture for the spoofing detection task with far fewer trainable parameters. Furthermore, we also investigate the effect of network parameterisation on performance in Subsection 4.3.4. Subsection 4.3.5 then provides a summary of the work done in this section.

It should be noted that all the models are trained and tested on the ASVspoof 2017 v1.0 dataset (see Section 3.2.1) and use the EER metric for performance evaluation (see Section 3.5). Next, we provide a short description of the published deep learning systems from the ASVspoof 2017 challenge evaluations.

Best published deep learning systems

Below we provide a summary of deep learning based systems that have been evaluated on the ASVspoof 2017 v1.0 dataset.

- System A [Lavrentyeva et al., 2017]: This system used score-level fusion of three sub-systems. The first is a GMM trained on features extracted from a CNN — which in turn is trained on log-power spectrograms. The second is an i-vector based SVM system where i-vectors were extracted from LPCC. And the third is an end-to-end CNN-RNN system trained on log-power spectrograms.

- System B [Nagarsheth et al., 2017]: The authors train a CNN as a feature extractor using tandem features — combining CQCCs with High Frequency Cepstral Coefficients (HFCCs[5]). They train this network in a multi-class setting to model different spoofing attack configurations seen in the training and development set. Then for every audio recording in the dataset they extract feature embeddings using the pretrained CNN. A binary SVM classifier is then trained on these embeddings to discriminate between the bonafide and spoofed classes.

- System C [Chen et al., 2017]: This system employed score-level fusion of three sub-systems. The first is a GMM trained on the CQCC features. The second and third systems are residual neural networks (ResNets) trained on the MFCC and CQCC features respectively.

- System D [Cai et al., 2017]: This system used score-level fusion of three sub-systems. The first is a GMM trained on CQCC features (the baseline model from the challenge). The second is also a GMM system trained on CQCCs but uses augmented data during training (for the spoofed GMM).

[5][Nagarsheth et al., 2017] apply a high pass filter with a cutoff frequency of 3500 Hz to discard fundamental and harmonic speech frequencies and extract HFCC features.

Table 4.5: Performance (EER %) of the best deep learning systems on the development and evaluation sets.

System	Dev	Eval
A [Lavrentyeva et al., 2017]	3.95	6.73
B [Nagarsheth et al., 2017]	7.6	11.5
C [Chen et al., 2017]	2.58	13.29
D [Cai et al., 2017]	3.52	16.39
E [Alluri et al., 2017]	2.21	17.82
LCNN [Lavrentyeva et al., 2017]	4.53	7.34

The authors apply different parametric reverberators and phase shifters to create simulated replay data. The third system is a residual neural network trained on spectrograms.

- System E [Alluri et al., 2017]: This system used score-level fusion of a GMM and a Bi-directional long short term memory network (BLSTM). The authors use their proposed Single Frequency Filter Cepstral Coefficients (SFFCC) based delta-features to train the GMM and BLSTM models.

- LCNN [Lavrentyeva et al., 2017]: This is one of the sub-systems of A that uses features extracted from a CNN to train a single-component GMM to model spoofed and bonafide classes. This stand-alone system has contributed the most among other sub-systems in system A reporting an EER of 7.34% on the evaluation set (Table 4.5).

Table 4.5 summarises the results of these systems on the development and evaluation sets. We observe a remarkable performance by system A, the state-of-the-art [Lavrentyeva et al., 2017]. The top two systems (A and B) show similar levels of generalisation[6] on the development and the evaluation sets which however contradicts with the performance shown by the other systems C, D and E.

Motivation

Usually the success of deep learning systems is attributed to the availability of large training data. However, within the context of the ASVspoof 2017 v1.0 (see Section 3.2.1), where the training data is significantly less than the test data, training a deep neural network to be able to achieve good generalisation can be challenging. Therefore we outline the following questions that motivate the work in this section.

[6]The gap between the EER on the development and the evaluation datasets is nearly the same.

- Using only the available training and development data, is it possible to train an end-to-end CNN that generalises well to the evaluation dataset?
- Why is there a huge performance gap between the development and evaluation datasets? Could one reason be due to the high imbalance in the number of utterances between the two above mentioned sets?
- Lastly, given such a small training data (about 2.22 hours), can we design a deep architecture with fewer trainable parameters that neither underfits nor overfits on training data?

This work tries to seek answers to the above questions and discusses the possible outcomes. The next section will describe our experiments towards replicating the LCNN model.

4.3.2 Replicating the state-of-the-art LCNN

Here we describe our efforts in replicating LCNN — the best performing CNN model of the ASVspoof 2017 challenge that reported an EER of 7.34% on the evaluation set. We train our replicated version of the LCNN model using the same input representation and network parameterisation [Lavrentyeva et al., 2017] with the following difference[7]: we use 2048 FFT points and 2048 window size with a 10 ms hop size to compute the spectrograms. Therefore, our input spectrogram is of shape 400×1025 instead of 400×864 as used in LCNN [Lavrentyeva et al., 2017], where 400 denotes time frames and 1025 number of frequency bins. We either truncate or copy the original samples depending upon the original audio duration to create a 4 seconds fixed duration audio representation. We do this for every audio recording in the dataset. We use the Librosa[8] library for computing the spectrograms. Appendix A provides the details of the LCNN architecture.

Model training and testing

The input to the network is a mean-variance normalised log power magnitude spectrogram. We initialise our network weights using Xavier initialisation [Glorot and Bengio, 2010] and biases with zero. The network is trained to optimise the cross entropy loss between the bonafide and spoofed classes. As specified in [Lavrentyeva et al., 2017], we use max-feature-map (MFM) non-linearity, a learning rate of 0.0001, 32 as batch size, and 0.9 as the momentum. The network

[7]Our preliminary experiments using 1728 FFT points show worse performance on the evaluation set. We envisage that use of higher frequency resolution (2048 point FFT) should help improve detection performance.

[8]http://librosa.github.io

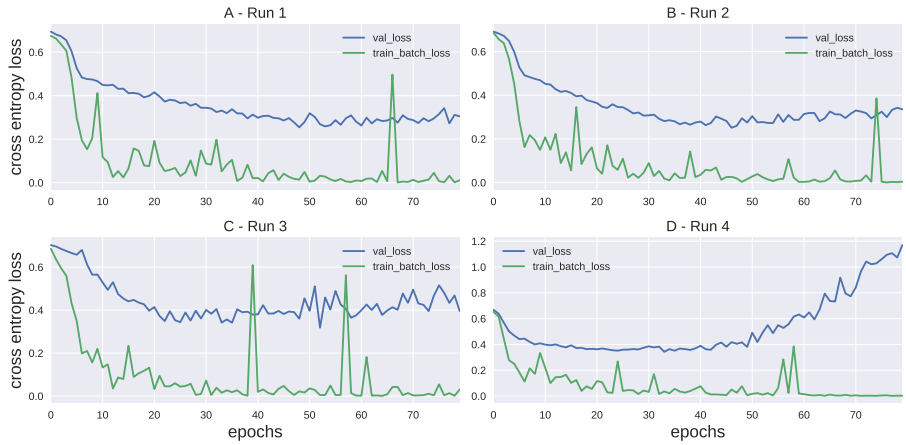


Figure 4.2: Cross entropy loss on the training and development sets for four different runs of training our replicated version of the LCNN.

is trained using stochastic gradient descent with the ADAM optimiser [Kingma and Ba, 2014]. The default parameter value of epsilon did not work and we use 0.1 instead. A dropout of 70% to the inputs of the first fully connected layer is used during model training. We use the tensorflow [Abadi et al., 2015] framework for the CNN implementation. We use early stopping[9] during training to overcome overfitting. If the validation loss does not improve for 30 epochs then we abort the training. We use a maximum of 300 training epochs and chose the model that shows the best performance on the validation data. At inference time, for each audio spectrogram the model outputs a posterior probability distribution for the bonafide and spoofed classes. The final score is obtained by converting these posteriors into a log likelihood ratio and the EER is computed. Using the above described approach, we train 5 different CNN models initialised with different random weights, with an aim to study how model behaviour varies across different runs of model training.

Results

Figure 4.2 shows the training and validation cross entropy loss visualisations for four different training runs of our replicated version of LCNN model. As expected, we see a decrease in the training loss as the number of epochs increase. Except for a few bumps in some training epochs the loss decreases smoothly reaching almost 0% after few training epochs. This holds true for all four training runs. However, we do not observe much improvement in the validation loss after 20 - 30 epochs suggesting a case of overfitting. All our

[9]Lavrentyeva et al. [2017] do not mention using early stopping in their paper.

Table 4.6: Performance (EER%) of our replicated CNNs in an end-to-end setting and training GMMs on CNN features. Shown results are for five different training runs.

	End-to-End		GMM	
	Dev	Eval	Dev	Eval
Run 1	9.04	32.02	9.49	34.00
Run 2	9.30	37.67	10.46	39.00
Run 3	8.01	30.96	9.40	34.24
Run 4	14.11	36.97	12.80	38.70
Run 5	9.11	37.34	10.78	35.66

models are trained on the training set and validated using the development set. We present the results of our replicated version of the LCNN model in Table 4.6. Furthermore, for comparison with the state-of-the-art performance we also build a GMM system trained on features extracted from our pre-trained CNNs. For each audio recording, a 32 dimensional feature vector, which is the output of the fully connected layer (with 32 units), is extracted. A one-component GMM is then trained to model the bonafide and spoofed feature distribution and the log-likelihood difference between the bonafide and spoof GMMs is used for scoring as described in Equation 4.1. Having run the setup for five different runs with different random initialisations, none of our replicated version models could achieve a performance even half of what Lavrentyeva et al. [2017] reported under the common training conditions. We observe the best performance for Run 3, with an EER of 8.01% and 30.96% on the development and evaluation sets under an end-to-end setting and 9.4% and 34.24% using GMMs. We also tried a small experiment altering the training and validation sets, where we now use the development set for learning CNN parameters and the training set for validation. Using this approach our model reported a small EER of 2% on the validation set (which is the training set in this case), but worse performance on the evaluation set.

We make the following observations: (1) We do not see a substantial difference in performance by training the GMM on CNN features in contrast to the end-to-end model, suggesting that the features learned by CNN are not generalisable; (2) On the evaluation set we see a large variance in performance across different runs of model training, suggesting difficulty in reproducing the same results; (3) The performance gap between the development set and evaluation set is always large. This suggests overfitting on the development set and that the development set partition may not be representative of the unseen evaluation set.

Therefore these experiments suggest that it is quite difficult to replicate the LCNN model to achieve the same level of generalisation between the develop-

ment and evaluation sets using the limited details provided by Lavrentyeva et al. [2017]. In the next section we investigate new CNN architectures with an aim to achieve better generalisation on both the development and the evaluation sets.

4.3.3 Investigating alternative CNN architectures

This section now describes three different CNN architectures using log-power spectrogram inputs for replay spoofing attack detection. The first architecture which we call Model_1, is adapted from [Nagarsheth et al., 2017] who reported the second best performance using CNNs on the ASVspoof 2017 v1.0 evaluation set. The second architecture, which we call it Model_2, is adapted from [Grill and Schlüter, 2017] which is the best performing model of the Bird Audio Detection (BAD) challenge 2017. The third architecture which we call Model_3 is our proposed CNN architecture having a smaller number of model parameters yet reports comparable performance with all other CNN models studied here.

$\textbf{Model}_1$

This section now describes the methodology used to train Model_1, which uses the CNN architecture of the second best performing CNN-based model [Nagarsheth et al., 2017] of the ASVspoof 2017 evaluations. As mentioned earlier in Section 4.3.1, the authors trained a CNN on tandem features (CQCCs + HFCCs) to model different spoofing attack configurations in the training (and development) set in a multi-class setting. Later the authors used this pretrained CNN for feature extraction and trained an SVM classifier for spoofing detection. In our case, we adopt this architecture but use two output targets to model the bonafide and spoofed classes in an end-to-end setting using log-power spectrograms as its input representation. Furthermore, inspired by Nagarsheth et al. [2017], we only chose to use the first one second of audio during training and evaluation. As described earlier in Section 4.3.2, we copy or truncate samples to obtain a one second audio representation for every audio recording. In this setup we use a 512 point FFT, 512 window size and 10 ms hop size to produce a unified input spectrogram of 100×257 (time x frequency) dimensions.

The network comprises three convolutional layers with 128 filters each. The first convolutional layer uses a 3×257 (time x frequency) filter while the second and third layers use 3×1 size filters. All the convolutional layers use a stride of 1×1. This is followed by a max pooling procedure over the time axis. The pooled input is then fed to a series of three fully connected (FC) layers each having 256 units. As all the implementation details of their CNN are not disclosed in [Nagarsheth et al., 2017], we chose to use the parameter initialisation and training approach described in Section 4.3.2.

Table 4.7: Performance comparison of different CNN architectures studied.

System	Dev EER (%)	Eval EER (%)	# params
LCNN (our)	8.01	30.96	371 K
Model_1	5.47	25.28	4 M
Model_2	4.52	34.91	68 K
Model_3	4.98	33.11	7682

The performance of Model_1 in terms of EER% is shown in Table 4.7. Here we only report the best performing model obtained using this architecture which shows 5.47% and 25.28% EER on the development and evaluation data. However, this model uses a high dropout rate: 90% and 80% to the inputs of the first and second FC layers and 60% to the inputs of the output layer.

Model_2

This CNN architecture is motivated from the work of Grill and Schlüter [2017] that was submitted to the 2017 Bird Audio Detection (BAD) challenge. Although the objectives of the BAD and ASVspoof 2017 challenges are completely different, they exhibit some similarities in the proposed test conditions: both focus on wild and diverse test conditions. Therefore, we adopt one of their CNN architecture called "Bulbul" to study if changing the architecture helps improve the performance gap (better generalisation) between the development and the evaluation sets applied to the replay spoofing detection task. Our adapted network has four convolutional layers each having 16 kernels/filters. Each convolutional layer is followed by a max pooling layer. The first two convolutional layers use a 3×3 filter with a stride of 1×1 while the last two convolutional layers use a 3×1 size filter with 1×1 stride. The first two pooling layers use a 3×3 size filter and stride, and the last two pooling layers use a 3×1 size filter and stride. This is followed by two fully connected layers consisting of 256 and 32 units. Unlike the single unit used by [Grill and Schlüter, 2017] in the output layer we use two units to model the probability distribution for bonafide and spoofed classes.

In this setup, partly motivated by [Grill and Schlüter, 2017], we create excerpts of 1 second duration from the original audio files and use them during training and testing. For a given audio utterance s, we use the following algorithm to split the original data and prepare excerpts of 1 second duration.

1. Let $l = length(s)$, be the original duration of s. Update s by copying or truncating the original samples such that $l_{new} = ceil(l)$.

2. Compute the log power magnitude spectrogram: $D = \log\ |STFT(s)|^2$ using a 256 point FFT, 256 window size and a 10 ms hop size. Here, the

resulting matrix D contains T time frames and F number of frequency bins.

3. Let $spec_{wind}$ and $wind_{shift}$ be the desired excerpt (window) size and window shift size (in time) respectively. Now split D into different excerpts by moving $spec_{wind}$ by $wind_{shift}$.

4. Return the list of spectrograms generated in previous step, where each spectrogram is of dimension $spec_{wind}$ x F.

Using the above outlined steps and 1 second (100 frames) $spec_{wind}$ and $wind_{shift}$ we created spectrogram excerpts of 1 second duration and used them during training and testing. Therefore, the input to the network in this setup is a spectrogram of 100×129 dimensions. At test time we take the average of the scores obtained for different spectrogram excerpts corresponding to a given test utterance and compute the EER. We use the parameterisation and training recipe as described in Section 4.3.2.

The performance of Model_2 is shown in Table 4.7. Although we experimented with different dropouts, we found the best result using 50% dropout to the inputs of the last two FC layers and 90% to the first FC layer inputs. This model gives 4.52% and 34.91% EER on the development and the evaluation sets respectively. However, the generalisation gap between the two test sets is large, suggesting that our model might be overfitting on the development set.

Model_3

Our work so far in this section has investigated different CNN architectures. These architectures range from medium to complex in terms of trainable parameters of the network (see the last column of Table 4.7). However, none of these systems showed a similar level of generalisation on the development and evaluation sets of the ASVspoof 2017 v1.0 dataset. Therefore, we now propose an architecture with a smaller number of parameters which shows comparable performance with other network architectures studied so far. This architecture, partly motivated from the previous two architectures (Model_1 and Model_2), has three convolutional layers and two fully connected layers. Each convolutional layer has 16 output filters (feature maps) and uses a small rectangular filter of size 1×9 with a stride of 1×1 along the time and frequency axes. We apply a max pooling operation after each convolution layer. We use a 3×3 kernel and a stride of 3×3 in all max pooling layers. We use 32 neurons in the first FC layer with linear activation and two neurons in the output layer. All other layers use MFM non-linearity (described in Subsection 2.6.2). We apply a 50% dropout to the first FC layer inputs. Our proposed network topology is shown

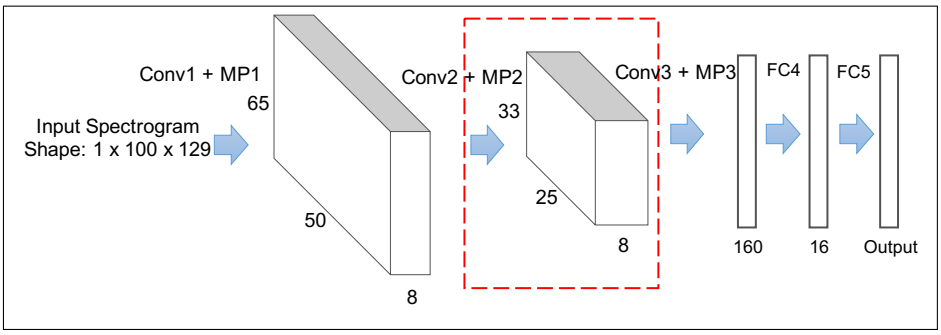


Figure 4.3: Architecture of the proposed model. The highlighted component shows a layer and its output feature map. For example, the shape of the feature map after the second convolutional layer and max pooling layer is 33 x 25 x 8. FC: fully connected, MP: max pooling.

Table 4.8: Model_3 performance (EER %) for different activation functions.

Activation function	Dev	Eval
MFM	**4.98**	33.11
ReLU	5.29	**31.7**
ELU	8.66	40.78

in Fig. 4.3. The input representation, model training and testing approach used for Model_3 is the same as in Model_2 described earlier.

The performance of Model_3 is shown in Table 4.7. Our proposed architecture appears to work quite well giving about 5% EER on the development set. However, our model shows a worse generalisation on the evaluation set yielding an EER more than 30%.

4.3.4 Effect of parameterisation on performance

Now we take our Model_3 which showed comparable performance with other models we studied and investigate how different network parameterisations affect its performance. To this end, we look at how different choices of activation functions (non linearity) and mini batch size used during stochastic gradient descent optimisation affect the model performance. One key motivation to do this is that so far we have used some of these parameterisations directly from [Lavrentyeva et al., 2017] considering the impressive performance they reported during the ASVspoof 2017 evaluations.

Activation function vs EER

All our models studied so far have used the MFM activation function in both convolutional and fully connected layers following [Lavrentyeva et al., 2017].

Table 4.9: $Model_3$ performance (EER %) for different batch sizes.

Batch size	Dev	Eval
8	**4.62**	36.02
16	5.64	35.35
32	4.98	**33.11**
64	5.96	36.6

Table 4.10: $Model_3$ performance (EER %) using a single spectrogram excerpt and multiple one-second excerpts per audio file.

spectrogram	Dev	Eval
split	4.98	33.11
single	6.5	35.56

Here, we compare the performance of MFM with two other popular activation functions: ReLU and ELU, that are often used in various deep learning tasks. Subsection 2.6.2 provides necessary background on these activation functions. $Model_3$ is now trained and tested using the ReLU and ELU activation functions using the same training and testing approach described earlier in Subsection 4.3.3. Table 4.8 summarises the results. The ELU activation shows worse performance on both the development and evaluation sets. Although ReLU shows competitive performance with MFM on the development set, it yields the best performance on the evaluation set outperforming both MFM and ELU activations.

Batch size vs EER

Following [Lavrentyeva et al., 2017], we have used a mini batch size of 32 samples in training all the CNN models in this section. Using the $Model_3$ architecture, we now investigate how performance compares when the network is trained using different mini batch sizes. To this end, we experiment with 8, 16 and 64 batch sizes. We use the same training and testing approach as described earlier in Subsection 4.3.3. We present the results in Table 4.9. We see worse performance on both the development and evaluation sets for a 64 batch size. Similarly, batch size 16 does not seem to work well. Overall, we see an optimal performance for a batch size of 32.

Split data vs EER

A final set of experiments we perform on $Model_3$ is to see the variation in the performance when the same model is trained on one-spectrogram excerpt (of 3 seconds duration) and multiple spectrogram excerpts of 1-second durations (obtained using the approach described earlier in Subsection 4.3.3). For the

one-spectrogram representation, we use three seconds audio obtained either by truncating or copying the original audio samples to match the duration. Therefore, in this setup the input spectrogram per each audio file is of shape 300×129 with 300 time frames and 129 frequency bins. We used 256 point FFT, window size of 256 and 10 ms hop size. We train and evaluate this model using the same method described in Subsection 4.3.3. Table 4.10 summarises the results. The model trained on split spectrograms outperforms the single-spectrogram representation model on both the development and evaluation sets. However, the gain in performance is not substantially large as we would expect by increasing the training data points where the original spectrogram is split into multiple excerpts.

4.3.5 Discussion

This section described our efforts towards replicating the state-of-the-art LCNN model that showed the best replay spoofing attack detection performance in the ASVspoof 2017 challenge evaluation set. Our experimental results suggest that using only the available audio examples in the training set along with the details provided in their paper [Lavrentyeva et al., 2017] it is not easy to replicate their model to reach even $10 - 15\%$ EER on the evaluation set. Following this, we investigated alternative CNN architectures with an aim to see if the performance on the evaluation set can be improved. To this end, we also proposed a CNN architecture with fewer trainable parameters that showed a similar level of performance as other models with more parameters. In all our experiments the trained models failed to generalise on the evaluation set but achieved good performance on the development set. We find that despite trying different CNN architectures it is very easy to overfit not only on the training set but overfitting on the validation set (the development set is always used for model validation) is quite prevalent. Hence, reducing the performance gap between the development and evaluation sets is challenging. Taking our proposed Model 3 architecture, we then studied the effect of network parameterisation: activation functions and mini batch sizes, on the model performance. Experimental results suggest that 32 is an optimal choice for mini batch size on this dataset. Interestingly, we find that the ReLU activation function shows superior performance over the MFM activation (proposed by Lavrentyeva et al. [2017]) on the evaluation set. Finally, our proposed idea of splitting the original data (spectrogram) into different splits (excerpts) resulting in increased training data points — a kind of data augmentation strategy — did not show notable performance gains in contrast to the model trained with a fixed-duration single spectrogram input representation (see Table 4.10).

4.4 Analysing spoofing countermeasure performance under varied conditions

4.4.1 Introduction

As highlighted in Sections 2.3 and 3.2, the ASVspoof 2017 dataset contains spoofed utterances created under 'wild' replay attack conditions. Although Lavrentyeva et al. [2017] reported an EER of about 6% on the evaluation set of the v1.0 ASVspoof 2017 dataset, the number is still high, indicating difficultly towards reliable detection of replay spoofing attacks under wild conditions on this dataset. Furthermore, Section 4.3 demonstrated that it is also not easy to replicate the LCNN model of [Lavrentyeva et al., 2017] using details published in their paper. Following these findings, this section aims at analysing the performance of countermeasure models across different replay attack conditions in the evaluation set. For this, both frame-level and utterance-level countermeasure models are investigated. Following the potential issues identified in v1.0 of the ASVspoof 2017 dataset, an updated version (v2.0) of the dataset was released by the ASVspoof challenge organisers (see Section 3.2). From hereon, all the work reported on ASVspoof 2017 dataset in this thesis uses v2.0 of the dataset.

To this end, Subsection 4.4.2 provides details of the experimental design considered here. The description of CQCC feature-based GMM baselines, CNNs trained on spectrogram inputs, GMMs and SVMs trained on hand-designed features and various ensemble models is provided. Subsection 4.4.3 then evaluates the performance of these countermeasure models using the EER metric (as described in Subsection 3.5.1). Subsection 4.4.4 then analyses the performance of these countermeasures under varied replay attack conditions. First, it studies how EER varies across different qualities of playback devices, recording devices and acoustic environments used to derive replayed recordings in the ASVspoof 2017 evaluation set. It then looks at how different qualities of replay attack configurations impact countermeasure performance. Three different qualities of attacks: low, medium and high as defined by Delgado et al. [2018] are used in this study. The section then analyses two of the GMM systems for both low and high quality replay configurations. It looks at how frame-wise energy and log-likelihood scores are distributed across frames with an aim to understand what influences model decisions. Finally, Subsection 4.4.5 provides a summary of the work done in this section. This work was published in Chettri et al. [2018c].

4.4.2 Experimental design

This section discusses the baselines and different countermeasure models considered in this study: (1) CNNs trained on spectrogram inputs; (2) GMMs trained on MFCCs and IMFCCs; (3) SVMs trained on MFCC i-vectors and IMFCC i-vectors; and finally (4) Fusion systems. We use pooled (train+development) data for training all our systems except for the CNNs that are trained on the training set, with the development set being used for model validation. As for performance evaluation, this section uses the EER metric (described in Section 3.5.1) for all models.

Baselines

The baseline system used during the ASVspoof 2017 evaluations on v1.0 of the dataset was a 512 component GMM trained on 90 dimensional CQCC features that was obtained by combining the delta and acceleration coefficients of the first 30 static coefficients [Kinnunen et al., 2017a]. Using the same feature configuration we train this baseline but on v2.0 of the dataset. We call this system B_1. We also design another baseline system B_2 using the feature parameterisation from Delgado et al. [2018] on v2.0 of the dataset. Although B_2 is also designed using 512 mixture components on CQCC acoustic features, the feature parameterisation is different. Every feature vector is now represented by a 60-dimensional vector obtained by taking the log energy (replacing the 0^{th} coefficients) and the first 19 static coefficients followed by their delta and acceleration coefficients. This is followed by cepstral mean variance normalisation. Although Delgado et al. [2018] calls this an enhanced baseline, this section refers to this system as B_2.

CNN-based systems

Two CNN-based countermeasure models, CNN_1 and CNN_2, are trained using spectrogram inputs. CNN_1 uses the architecture of our replicated LCNN model from Subsection 4.3.2. CNN_2 uses our proposed model architecture ($Model_3$) from Subsection 4.3.3. However, we make the following modifications in terms of input to these models. As the average duration of the training set in v2.0 of the ASVspoof 2017 dataset is about 2.66 seconds, we chose to use a 3 seconds fixed duration representation during training and testing. We copy or truncate the original audio samples to obtain this fixed duration input representation. Following our prior work in Subsection 4.3.3, we use a 256 point FFT, 256 window size and a hop of 10 ms for computing spectrograms. This holds true for both the CNNs. Thus, our input to the CNNs is a mean-variance normalised log power spectrogram of 300×129 (time $\times$ frequency) dimensions,

where time denotes the number of frames and frequency the number of bins. The means and variances per frequency bins are computed from the training set. Furthermore, in both the CNNs we use the ReLU activation function in both the convolutional and fully connected layers following our findings in Subsection 4.3.4. However, we use the same training and testing methodology as described in Subsection 4.3.2 for training our replicated version of the LCNN model.

GMM-based systems

Following our prior work in Subsection 4.2.2, we chose two standard short-time spectral features, MFCCs and IMFCCs, and build two GMM countermeasure models, GMM_1 and GMM_2, respectively. Feature extraction, training and testing of these GMMs is the same as described in Subsection 4.2.2. The score is computed as the log likelihood ratio between the bonafide and spoofed GMM models as described in Equation 4.1. GMM_1 uses 512 mixture components trained on MFCCs and GMM_2 is trained on IMFCCs using 256 mixture components. Both GMMs use a 40-dimensional feature vector per frame obtained by concatenating delta and acceleration (DA) coefficients. This choice of feature parameterisation comes from our prior findings in Subsection 4.2.2.

SVM-based systems

We train two utterance-level binary SVM models, SVM_1 and SVM_2, using i-vectors [Dehak et al., 2011]. SVM_1 is trained using MFCC feature-based i-vectors and SVM_2 uses IMFCC feature-based i-vectors. We use the same 40 dimensional DA feature representation for both MFCCs and IMFCCs that was used in training the GMMs. Using this feature representation we train the total variability matrix (T-matrix) and universal background model (UBM). We use the pooled (train+development) data to train the UBM with 128 mixture components and the T-matrix with a 100 rank. We extract i-vectors for the entire dataset for both MFCCs and IMFCCs. Then, the SVM_1 and SVM_2 models with a linear kernel are trained on pooled i-vectors (MFCC and IMFCC based i-vectors respectively) to discriminate between the bonafide and spoofed classes. SVMs are implemented using Scikit-learn [Pedregosa et al., 2011] and the i-vector extractor is implemented using the MSR-Identity toolkit [Sadjadi et al., 2013].

Ensemble systems

We argue that a single feature and a single classifier may not adequately model the diverse spoofing conditions that appear in the ASVspoof 2017 evaluation set.

Table 4.11: Performance of countermeasure models on the evaluation set of the ASVspoof 2017 v2.0 dataset.

ID	System	Features	EER %
1	B_1	90 dimensional CQCCs	23.4
	B_2	60 dimensional CQCCs	12.2
2	CNN_1	Spectrograms	28.2
	CNN_2		27.8
3	GMM_1	MFCCs	27.8
	GMM_2	IMFCCs	18.3
4	SVM_1	i-vectors derived from MFCCs	24.6
	SVM_2	i-vectors derived from IMFCCs	16.3
5	$Fused_1$	Scores from systems in 2-4	12.3
	$Fused_2$	GMM_1+GMM_2 system scores	15.9
	$Fused_3$	SVM_1+SVM_2 system scores	11.5
	$Fused_4$	Scores from systems in $3-4$	**11.0**

To this end, we investigate detection performance using ensemble approaches which have shown promising results in both the ASVspoof 2017 [Lavrentyeva et al., 2017] and ASVspoof 2015 [Patel and Patil, 2015] challenges. Therefore, we design four score-level fusion[10] models using the linear logistic regression implementation of Brümmer and de Villiers [2013]. $Fused_1$ combines scores from all the six countermeasures: CNN_1, CNN_2, GMM_1, GMM_2, SVM_1 and SVM_2. $Fused_2$ combines the GMM_1 and GMM_2 scores while $Fused_3$ fuses the scores of the SVM_1 and SVM_2 models. Finally, $Fused_4$ combines the two GMMs and the two SVMs.

It should be noted that in our work we consider CQCC-based GMMs as baselines to compare the performance of our systems. Therefore we do not include them in our fusion setups, although including them may offer a gain in performance.

4.4.3 Evaluation

We now evaluate the performance of all our countermeasure models on the ASVspoof 2017 v2.0 evaluation set using the EER metric. Table 4.11 summarises the results. The two baseline GMMs B_1 and B_2 produce an EER of 23.4% and 12.2% respectively. The end-to-end CNN models CNN_1 and CNN_2 show poor performance on the evaluation set. CNN_1 shows an EER of 28.2% and CNN_2 yields an EER of 27.81%. A possible reason for the poor generalisation of both CNN models might be attributed to the small amount (only 2.22 hours) of data available for training the models. GMM_2 trained on IMFCCs shows an

[10]We explore many different model combinations but report only the four ensemble systems that showed best performance on the evaluation set.

EER of 18.3%, clearly outperforming GMM_1 which is trained on MFCC features with an EER of 27.8%. We find a similar trend in performance for the i-vector based SVM models. SVM_2, which is trained on IMFCC-based i-vectors, shows an EER of 16.3%, outperforming the MFCC-based i-vector SVM_1 model with 24.6% EER. Thus, IMFCCs that emphasise higher frequencies of the speech signal seem to give better performance over MFCCs in general.

By now, it is quite evident how hard it is for a single countermeasure to counter the diverse nature of replay attacks in the evaluation set. Thus, we investigate the benefits that these countermeasures offer as an ensemble system. The first ensemble system $Fused_1$ produces an EER of 12.3%, offering about 11% absolute gain over the baseline B_1 and a comparable performance with the baseline B_2. The $Fused_2$ model shows an EER of 15.9% and the $Fused_3$ model an EER of 11.5%. Our best fusion system $Fused_4$ reports an EER of 11.0%, outperforming both the baselines B_1 and B_2 by 13.4% and 1.2% absolute value, respectively. These results suggest that ensemble approaches could be one possible direction for further investigation.

The results seen so far do not explain what these models have learned to make predictions or which factor (among acoustic environment, playback and recording device in a replay attack) influences the prediction most. Thus, we perform an analysis on countermeasure performance for different replay spoofing conditions in the next section.

4.4.4 Analysis

This section describes the performance analysis of the various countermeasure models considered in this study in terms of different factors of a replay attack. The details of different factors such as acoustic environment, playback devices and recording devices used in creating replayed utterances of the ASVspoof 2017 v2.0 dataset are provided in Subsection 3.2.2. This section first aims at understanding the impact on model performance for different qualities of playback and recording devices and the type of acoustic environment individually. Then it aims at analysing how the performance of countermeasures varies across different qualities of replay configurations (RC) as a whole. To this end, three different configurations of low, medium and high qualities are investigated. For more details on different RCs please see [Delgado et al., 2018]. Furthermore, we do not use all our countermeasure models for this analysis. We choose B_2 and CNN_2 as they showed better results over the B_1 and CNN_1 models. We also include both GMMs and SVMs in our analysis. Among different fusion models, we choose the $Fused_2$ model in this analysis. One motivation for this is that $Fused_2$, which combines GMM_1 and GMM_2, shows a substantial perfor-

mance improvement over the GMM_1 model trained on MFCCs (EER of 27.8%). Finally, this section also performs frame-level analysis using one low- and one high-quality replay configuration in the evaluation set. For this, the GMM_1 and GMM_2 models are used.

Impact of device quality and acoustic environment

To study the impact on model performance for different quality of recording devices, playback devices and acoustic environments, following Delgado et al. [2018] we pool all the evaluation set scores according to low, medium and high quality categories for each factors: acoustic environment (AE), playback device (PD) and recording device (RD). We then evaluate the performance in terms of EER for each of these categories. Table 4.12 summarises the results. Note that the number of replayed utterances varies across different conditions (eg. 9336 for medium quality and 1633 for high quality acoustic environments) but the number of genuine utterances remains the same[11]. We make the following observations. (1) The $Fused_2$ model outperforms the baseline B_2 under each category except for low quality recording devices, indicating that ensemble approaches do help improve detection performance. (2) CNN_2 shows worse performance compared to the baseline B_2. (3) Generally, SVM models tend to show better performance over GMMs (with few exceptions in some cases). (4) AE: for the low quality AE, the MFCC feature-based models (GMM_1 and SVM_1) show better performance over IMFCC feature-based models (GMM_2 and SVM_2). However, we see an opposite trend for replay attacks using medium and high quality AE. (5) PD: for the low quality PD, the MFCC feature-based i-vector model SVM_1 outperforms the IMFCC feature-based model SVM_2, but on the medium and high quality PD SVM_2 outperforms SVM_1. (6) RD: for all low, medium and high quality RDs, the IMFCC-ivector based SVM_2 model outperforms SVM_1 based on MFCC i-vectors.

The experiments conducted here make an assumption that while analysing the influence of one factor, say AE, the other two factors PD and RD have negligible impact. This however is not true because it is difficult to mask out the information related to RD and PD from a replayed audio signal. Had this been true, the problem of replay attack detection would have been less difficult to solve. Therefore, we argue that the results reported here may not be completely insightful to understand a replay spoofing detection system. This leads us to perform an analysis according to different qualities of replay configurations in the next section.

[11]The evaluation subset has 1298 genuine utterances. Whether we compute EER for the low AE or high AE the number of genuine utterances is always 1298.

Table 4.12: Performance (EER%) of countermeasures across different qualities of acoustic environments (AE), playback devices (PD) and recording devices (RD). Shown results are on the ASVspoof 2017 v2.0 evaluation set. Bold indicates systems outperforming the baseline B_2.

Replay factor	Quality	B_2	$Fused_2$	CNN_2	GMM_1	GMM_2	SVM_1	SVM_2
Acoustic environment	**Low**	16.6	**13.3**	24.2	**13.5**	18.8	**15.5**	20.4
	Medium	18.7	**11.4**	29.2	25.4	**18.5**	22.2	**16.2**
	High	21.8	**14.3**	22.3	44.4	**16.9**	41.3	**13.5**
Playback device	**Low**	16.6	**12.9**	36.4	18.3	21.9	**10.5**	21.6
	Medium	16.4	**9.9**	23.9	**11.9**	**16.2**	16.4	**11.8**
	High	18.3	**11.6**	20.7	35.7	**14.5**	34.8	**11.7**
Recording device	**Low**	10.8	12.2	27.9	22.5	18.4	21.4	17.6
	Medium	15.6	**10.0**	38.6	36.3	16.4	21.7	**12.7**
	High	17.7	**12.3**	24.5	28.7	18.8	28.2	**15.9**

Impact of different replay configurations

This section aims at understanding the impact of different qualities of replay attacks (or replay configurations) on the countermeasure models. We consider three low quality RCs: RC15 (E02 P21 R18), RC16 (E02 P21 R14) and RC19 (E02 P20 R14); three medium quality RCs: RC30 (E15 P19 R20), RC33 (E13 P14 R04), RC34 (E17 P12 R04); and three high quality RCs: RC55 (E26 P24 R24), RC56 (E25 P13 R08) and RC57 (E24 P23 R23). The letters E,P,R denotes the environment, playback device and replay device IDs. Their details are provided in Tables 3.4, 3.5 and 3.6 of Section 3.2.2. We follow the same RC identifiers as defined by Delgado et al. [2018]. It is worth noting that these high quality RCs use analog wire acoustic conditions, meaning there is no physical sound propagation and hence are considered to be the most difficult replay attacks to be detected by a countermeasure. As in [Delgado et al., 2018], we pool all the evaluation set scores corresponding to these RCs and evaluate system performance. Table 4.13 summarises the results and the details of the RCs considered in this analysis.

In general, the Fused_2 model shows the best performance outperforming the baseline model B_2. **Low:** Under low quality RCs, we observe worse performance for CNN_2 and for the IMFCC feature-based GMM_2 and SVM_2 models. Though we expected IMFCC features to show better performance as they emphasise higher frequency regions which enable capturing ambient noise, we find contradictory results. The MFCC feature-based models show comparable performance with the enhanced baseline model B_2. For RC19, GMM_1 shows 7.0% EER which further reduces to 3.5% for the i-vector based model SVM_1, clearly outperforming the baseline (10.5%). **Medium:** except CNN_2, all other models show comparable or improved performance in comparison to the baseline. **High:** For high quality RCs, the MFCC feature-based models (GMM_1 and SVM_1) show worse performance indicating that low frequency information is not very helpful for discriminating high quality replay attacks. All other models including CNN_2 clearly outperform the baseline for the high quality RCs we investigated. On the RC55 configuration, the GMM_2 model shows a good performance of 3.8% EER, in comparison to the baseline (15.0%) which further reduces to 3.5% for SVM_2 using IMFCC i-vectors.

Overall, we make the following observations. Within the context of the ASVspoof 2017 2.0 dataset, (1) MFCCs seem to show better performance for low and medium quality replay attacks. IMFCCs on the contrary show poor performance in general. This suggests that information at low frequencies is helpful for detecting low quality attacks. (2) For the high quality category (the hardest ones), MFCCs show the worse performance while IMFCC feature-based

Table 4.13: Performance (EER%) of countermeasures across different replay attack configurations (RCs) in the evaluation set of the ASVspoof 2017 v2.0 dataset. E, P and R in the second column represents environment, playback and recording device used to simulate replay attack. Bold represents systems outperforming the baseline B_2. The IDs in the first column are taken from [Delgado et al., 2018].

ID	Replay configuration	Quality	B_2	$Fused_2$	CNN_2	GMM_1	GMM_2	SVM_1	SVM_2
RC15	E02 P21 R18		8.0	10.7	19.9	**8.0**	23.2	13.6	24.2
RC16	E02 P21 R14	**Low**	9.0	**6.6**	21.4	12.5	13.9	13.3	18.0
RC19	E02 P20 R14		10.5	**8.5**	49.9	**7.0**	23.0	**3.5**	26.0
RC30	E15 P19 R20		7.0	**1.9**	16.0	**5.5**	7.1	**6.1**	**4.1**
RC33	E13 P14 R04	**Medium**	8.5	**5.0**	12.3	**6.4**	8.9	10.5	**7.9**
RC34	E17 P12 R04		9.0	**4.3**	12.5	**5.8**	**8.5**	10.2	**7.5**
RC55	E26 P24 R24		15.0	**11.0**	**9.8**	42.9	**3.8**	47.0	**3.5**
RC56	E25 P13 R08	**High**	36.0	**29.2**	**22.5**	43.4	**26.5**	48.1	**22.1**
RC57	E24 P23 R23		33.0	**27.4**	**26.6**	44.3	**26.4**	49.6	**22.3**

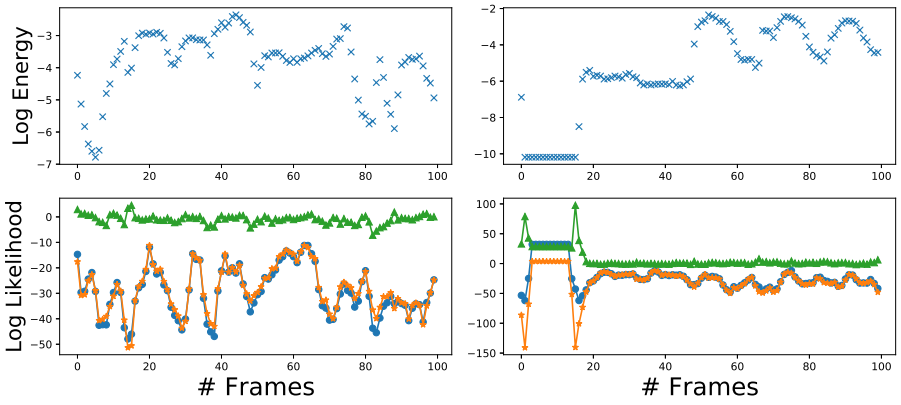


Figure 4.4: Log energy (top) and log-likelihood (bottom) plots for the first 100 frames of confidently classified spoofed (left) and bonafide (right) audio files in the evaluation set of the ASVspoof 2017 v2.0 dataset. Green: log-likelihood difference. Blue: bonafide GMM_2. Orange: spoofed GMM_2.

models show better performance, with SVM_2 trained on IMFCC i-vectors taking the lead. A possible explanation for this could be that these high quality devices may use low pass filters that mask out high frequency information in a replayed signal, leaving cues for discrimination. This hypothesis however needs further investigation that we aim to consider as our future work.

Frame-wise energy and log-likelihood analysis

Now we conduct frame-level analysis to see if we can derive any understanding about what the MFCC and IMFCC feature-based GMM models (GMM_1 and GMM_2) have learned about high quality replay attacks. For this we look at log energy and log-likelihood distributions across the frames of the most confidently classified spoofed and bonafide audio files under RC55[12].

Figure 4.4 shows the energy and log-likelihood distribution plots across the first 100 frames of bonafide and spoofed files for GMM_2. For the spoof file E_1005573.wav (left column of the figure), the energy distribution across frames seems to be uniform and smooth. The bonafide and spoofed model log-likelihood across the frames shows competitive behaviour, indicating how hard it is to have a clear boundary of discrimination between bonafide and replayed signals. Furthermore, we find that the log-likelihood difference (green profile) across the frames seems to be around zero indicating ambiguity in the decision boundary. However, on the bonafide file E_1002092.wav (right column of Figure 4.4), we find some cues about the bonafide class in the first few frames. We find lower en-

[12]Among the three high quality replay attacks/configurations we analysed (RC55, RC56, RC57) the IMFCC frontend shows the lowest EER for RC55, so we choose RC55 for analysis.

ergy for these frames in comparison to the remaining frames of the signal. Also, the spoof model for these frames assigns a very low likelihood score indicating that such frame instances were not seen during training for the spoofed GMM. As a result, the log likelihood ratio (green profile) in these frames dominates the other frames in the signal, thus serving as a key indicator favouring the bonafide class. We refer to these frames as outliers. Even for the MFCC feature-based GMM_1 model, we observe a similar trend as in Fig.4.4, and therefore do not include those plots here.

Table 4.14: Confusion matrix for GMM_1 and GMM_2 for low and high quality replay configurations: RC15 and RC55 respectively. **B**: bonafide, **S**: spoofed. Columns denote ground-truth and rows the predicted.

		RC15 (Low)		RC55 (High)	
		B	**S**	**B**	**S**
GMM_1 (MFCC)	**B**	1208	22	1208	162
	S	90	128	90	16
GMM_2 (IMFCC)	**B**	1197	116	1197	2
	S	101	34	101	176

From these observations, it appears that these models are also using the class-dependent data cues (outliers) found in the bonafide signals as one of the factors for making predictions. However, this would not be the case if a voice activity detector (VAD) was in place that would automatically eliminate non-speech frames, including such outliers. But this is not the case: these countermeasures use both speech and non-speech frames. Therefore, a realistic real-world replay countermeasure should detect and automatically handle such outliers during model training and testing to allow reliable prediction.

Finally, we look at the confusion matrices for GMM_1 and GMM_2 models for RC15 and RC55 conditions. One motivation to do this is that the reported EERs in Table 4.13 for different replay configurations do not provide significant statistical insights about the correct and incorrect classification for the bonafide and spoofed classes. The confusion matrices help understand the proportion of correct and incorrect classification. Table 4.14 shows this. For RC15, GMM_1 has high true negative (85.33%) but a small false positive (14.66%) rates while GMM_2 shows the opposite trend: high false positive (77.33%) and small true negative (22.66%) rates. On RC55, we see an opposite trend in contrast to RC15. Here, the GMM_1 model shows high false positive (91.01%) and low true negative (8.98%) while GMM_2 shows small false positive (1.12%) but high true negative (98.87%) rates. For genuine cases, both GMM_1 and GMM_2 show comparable performance (in terms of true positives and false negatives).

4.4.5 Discussion

This section investigated and analysed various countermeasures for replay spoofing detection on the ASVspoof 2017 v2.0 dataset. We find that the models using MFCC frontends have a smaller EER than the models using IMFCC frontends in the evaluation set when looking at replay configurations with supposed low quality. We find the opposite when looking at replay configurations with supposed high quality. However, gaining an in-depth understanding of what is causing this behaviour seems challenging because the original RedDots corpus [Lee et al., 2015] of bonafide recordings includes both clean and noisy recordings collected from heterogeneous devices, but lacks documentation on the meta-data (acoustic conditions, recording devices). This means that "high-quality spoofing conditions" may actually be low quality since the bonafide files were of low quality and vice-versa. Thus, on this dataset it is difficult to perform evaluation on factors (AE, PD, RD and RC) influencing replay attacks in controlled conditions and provide significant conclusions whether reverberation noise or some device-specific (recording or playback) attributes provide a cue to replay signal discrimination. Furthermore, our analysis suggests that the models may be using dataset-specific artefacts during prediction. Therefore, on this dataset, a reliable replay detector should automatically take care of such outliers and allow learning algorithms to exploit only the information related to replay factors to make a reliable prediction. This section also showed that ensemble models have potential towards improving detection performance. To this end, data augmentation approaches, augmented with ensemble techniques, may improve generalisation, and therefore should be investigated further.

4.5 Explaining CNN predictions

4.5.1 Introduction

As described in Subsection 4.3.1, the most successful systems in detecting replay spoofing attacks on the challenging ASVspoof 2017 test set are the ones based on deep neural networks (DNNs). Although these systems have shown promising results, what they have actually learned to do has not been answered; they are often used as a black-box. Is a system that appears to detect a spoofing attack actually working with attributes relevant to the problem, or is it merely a product of how a train/test dataset was constructed [Sturm, 2014]? For example, Section 4.2 demonstrated how a frame-based GMM countermeasure trained for replay spoofing detection exploited artefacts in the dataset (ASVspoof 2017 v1.0) to make class decisions. Similarly, Section 4.4 highlighted a similar issue for frame-based GMMs trained on the updated version (v2.0) of the dataset. Can we trust such a system "in the wild"? Answers to these questions can not only help improve the security of ASV systems, but also motivate new spoofing attacks, and improve training databases.

This section attempts to answer some of these questions for a deep CM model trained on v2.0 of the ASVspoof 2017 dataset. Following the lessons learnt from Section 4.3 we propose and build a new CNN model that shows good performance on the version 2.0 of the evaluation set and a comparable performance to the state-of-the-art LCNN model on version 1.0 of this dataset. Subsection 4.5.2 describes these details. There exist several methods to understand the global or local behaviour of deep models [Montavon et al., 2018]. Here, we use the SLIME [Mishra et al., 2017] algorithm to generate explanations for individual predictions. It is based on the LIME algorithm [Ribeiro et al., 2016], which is an acronym for Local Interpretable Model-Agnostic Explanations. Subsection 2.9.2 provides background on SLIME. Subsection 4.5.3 then provides explanations produced using this method highlighting temporal and spectral regions that the model weighs heavily to form its decisions for each class. The section describes explanations produced at both the instance-level and model-level (on the whole dataset). Our findings show that a decision of a recording being "spoofed" is weighted heavily by the information present in the first 400 milliseconds of the recording. It then appears that at least some of the attributes the model has learned come from peculiarities of the database, and not from the difference in channel characteristics one would expect in a replay attack. Next, Subsection 4.5.4 demonstrates the significance of our analysis in two ways. We show how to manipulate misclassified spoof recordings to be judged as "spoofed" by the model, thereby lowering its EER; and we

show how to manipulate spoofed recordings such that the model judges them as "bonafide", thereby dramatically raising the EER. Finally, Subsection 4.5.5 provides a summary of the work done in this section. This work was published in [Chettri et al., 2018a].

4.5.2 Experimental design and evaluation

This section describes details related to the experimental design and evaluation of our CNN countermeasure model for replay spoofing detection. It provides details of the dataset and the evaluation metric considered here. The architecture details of the CNN, model training and testing methodology are described. We follow the lesson learnt from our study in Section 4.3 (such as parameter initialisation, training and testing methods) in training CNNs and further introduce several changes that helped achieve performance closer to the LCNN model[13]. For comparative purposes, we train and evaluate our CNN on v1.0 of the dataset. The section then evaluates the performance of the reproduced models in terms of the EER metric.

Dataset and evaluation metric

This study uses the ASVspoof 2017 v2.0 dataset described in Subsection 3.2.2 and the EER metric for performance evaluation as described in Subsection 3.5.1.

CNN architecture and input representation

The architecture of our proposed CNN is adapted from the state-of-the-art LCNN [Lavrentyeva et al., 2017] model which showed the lowest EER on the evaluation set during the ASVspoof 2017 spoofing detection challenge. It comprises 5 convolution (Conv) layers, 4 network-in-network layers, 5 max-pooling layers and 2 fully connected (FC) layers. In other words, the structure in terms of stacking series of convolutional and fully connected layers remains the same as in LCNN. However, following our findings in Section 4.3, we now focus more on model generalisation. For this, we introduce following changes. First, we reduce the number of kernels in each convolutional (Conv) layer by a factor of 2 to keep the number of free parameters to a minimum. Second, we introduced a batch normalisation layer before applying non-linearity in every Conv layer to ensure features follow a normal distribution after the convolution operation. Third, we use 32 neurons in the first FC layer and a single neuron in the output layer in contrast to 64 neurons and two neurons used in the original LCNN.

[13]The author would like to thank Héctor Delgado (who was one of the organisers of the ASVspoof 2017 challenge), Senior Research Scientist at Nuance Communications Inc., for the fruitful discussion and help in developing the proposed CNN model.

Table 4.15: Performance (EER%) of our replicated LCNN models: M_1 and M_2 on the ASVspoof 2017 dataset respectively.

System	DB version	Train	Dev	Eval
LCNN [Lavrentyeva et al., 2017]	1.0	-	4.53	7.34
M_1	1.0	0.0	7.0	9.44
M_2	2.0	0.0	7.6	10.6

Furthermore, we now use a binary cross entropy loss instead of softmax loss in training CNNs. As for the non-linearity, we use the ReLU activation in both Conv and FC layers following our findings in Section 4.3. Our proposed CNN has 189k trainable parameters which is much smaller in comparison to the 572k parameters of the LCNN [Lavrentyeva et al., 2017]. Further details on the CNN architecture are provided in Appendix A.

The input to the network, as in LCNN, is a mean-variance normalised log power spectrogram of 4 seconds duration. We perform this normalisation at the utterance-level[14] to standardize the features within a given recording. Since the ASVspoof 2017 dataset uses 10 different phrases (shown in Table 3.1), the duration of audio files varies across these phrases. To obtain a consistent input representation we replicate[15] the audio samples if the duration is smaller than 4 seconds or truncate them if the duration is more than 4 seconds. Here, we use a 1728 point FFT, and a 108 ms window (1728 samples) with a hop size of 10 ms. Therefore, the input spectrogram has dimensions 865×400, where 865 is the number of frequency bins and 400 the number of time frames.

Training and testing

As for training the CNN model, we follow the same approach described in Subsection 4.3.2 with the following changes: the shape of the input spectrogram is now 865×400 instead of 1025×400; to avoid overfitting, we train our CNN for a maximum of 100 epochs (instead of 300) with 10 epochs for early stopping (instead of 30). We implement the model using the Keras [Chollet, 2015] framework with the TensorFlow backend. During testing, for each test utterance we use the model output — the posterior probability of being genuine — as the score and compute the EER. Using the above approach, we train two models M_1 and M_2 on versions 1.0 and 2.0 of the ASVspoof 2017 dataset, respectively.

Results

Table 4.15 summarises the performance of our replicated versions of the LCNN models: M_1 and M_2. Model M_1 shows comparable performance with the original LCNN model trained on v1.0 of the ASVspoof 2017 dataset. Further it should be noted that reproducing[16] the exact same results is difficult due to high dropouts and random weight initialisation used during training. Since our main objective in this section is to understand what the CNN has learned about spoofing detection, we do not focus on minimizing the EER of LCNN. From hereon, we only consider M_2 for further analysis as it uses the updated (v2.0) dataset.

4.5.3 Explaining predictions using SLIME

This section aims at explaining the predictions of the M_2 model using the SLIME algorithm that is described in Subsection 2.9.2. First, explanations are produced at instance-level taking the most confidently classified bonafide and spoofed instances in the training, development and evaluation sets. Both temporal and spectral explanations are generated for them. Following this, explanations are then produced for the entire dataset to derive a global model-level understanding of what M_2 has exploited from the underlying training data to make class decisions.

Instance-level explanations

We take the six most confidently correctly classified bonafide and spoofed audio instances from the training, development and evaluation sets and have SLIME generate explanations for their predictions. Table 4.16 summarises the weights assigned to each of the ten temporal segments for these six instances. The polarity of the learned weights signifies how the presence (or the absence) of a segment influences a model prediction. For example, the T8 and T5 segments in all the bonafide instances in Table 4.16 are segments in favour of and against the prediction, respectively. The bold numbers in the table represent the top two weighing segments/components. We refer to them as top_1 and top_2 explanations. For the genuine instances, SLIME assigns T8 and T9 as the top two explanations. We observe a marginal difference in the magnitude of weights assigned to T8 and T9 but a relatively larger difference for other temporal segments. These weights suggest that T8 and T9 offer more contribution towards

[14] Instead of computing means and variances from the training set as in Section 4.3.

[15] We replicate samples in the time domain.

[16] Reported models M_1 and M_2 are the best performing models on the development set among five different models trained with different random initialisations.

Table 4.16: Temporal explanations of the most confidently correctly classified audio instances in the training (T), development (D) and evaluation (E) sets. T1-T10 represent temporal segments in seconds. T1: 0-0.4, T2: 0.4-0.8, T3: 0.8-1.2, T4: 1.2-1.6, T5: 1.6-2.0, T6: 2.0-2.4, T7: 2.4-2.8, T8: 2.8-3.2, T9: 3.2-3.6, T10: 3.6-4.0. B and S denote the bonafide and spoofed classes. Bold values represent weights of the top two explanations.

			Weights assigned to different temporal segments									
Class	File	Probability(%)	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
B	T_1000780	87.8	0.009	-0.000	-0.002	0.007	-0.005	0.009	-0.003	**0.031**	**0.024**	-0.013
	D_1000260	91.2	0.007	-0.006	0.010	0.018	-0.012	0.005	0.001	**0.041**	**0.042**	-0.023
	E_1002535	88.4	-0.008	0.002	-0.000	0.006	-0.014	0.004	-0.000	**0.040**	**0.034**	-0.022
S	T_1002124	86	**0.335**	0.000	0.008	-0.000	0.008	-0.015	0.012	0.009	-0.005	**0.274**
	D_1001596	83.2	**0.507**	0.004	-0.013	-0.010	0.014	-0.007	0.007	0.006	-0.039	**0.119**
	E_1014008	81.0	**0.353**	-0.005	-0.015	0.010	0.009	-0.001	0.004	-0.018	-0.008	**0.207**

Table 4.17: Spectral explanations of the most confidently correctly classified audio instances in the training (T), development (D) and evaluation (E) sets. F1-F10 represent spectral bands in Hz. F1: 0-813, F2: 813-1626, F3: 1626-2439, F4: 2439-3252, F5: 3252-4065, F6: 4065-4878, F7: 4878-5691, F8: 5691-6504, F9: 6504 to 7318, F10: 7318 to 8000.

			Weights assigned to different spectral segments									
Class	File	Probability(%)	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
B	T_1000780	87.8	**0.019**	0.015	0.007	0.016	0.014	0.012	0.015	**0.019**	0.014	0.009
	D_1000260	91.2	0.018	0.018	0.020	0.013	0.017	**0.022**	**0.025**	0.011	0.020	0.013
	E_1002535	88.4	0.005	0.013	0.011	**0.016**	0.012	0.015	0.015	**0.018**	0.013	0.009
S	T_1002124	86	0.095	0.094	**0.108**	0.090	**0.105**	0.082	0.076	0.078	0.062	0.044
	D_1001596	83.2	0.094	**0.102**	0.091	0.100	0.091	0.093	**0.105**	0.097	0.080	0.055
	E_1014008	81.0	**0.136**	0.104	**0.145**	0.076	0.132	0.139	0.091	0.116	0.098	0.077

bonafide class decisions. For the spoofed instances, SLIME returns T1 and T10 as the top two explanations. Table 4.17 shows the spectral explanations for bonafide and spoofed class predictions on the same six instances used in Table 4.16. There is not much difference in the magnitude of weights across the spectral segments. For both the bonafide and spoofed decisions, it seems that M_2 uses information across most of the spectral bands.

Using only the explanations for few confidently classified audio instances would not provide a global understanding of the model behaviour. Therefore, in the next section we apply SLIME to every audio instance of the ASVspoof 2017 2.0 dataset and study the prediction explanation (weights) statistics to derive conclusions about what M_2 has learned to make predictions.

Model-level explanations

Now we apply the SLIME algorithm across the entire training, development and evaluation set. We record the top_1 explanations returned by SLIME and present the count statistics for top_1 explanations. This helps us derive a global understanding on the behaviour of M_2 for bonafide and spoofed class decisions.

In the training set, M_2 correctly classifies 1505 out of 1507 bonafide instances with more than 70% confidence. Table 4.18 (first row) shows the temporal and spectral explanations on these 1505 bonafide instances. The temporal explanations suggest that though the majority vote for top_1 appears to be T9, other temporal components also have some contribution in the prediction. To summarise, we observe that the first four (T1-T4) and the last three temporal segments (T8-T10) contribute most towards the bonafide class decision. One possible reason for such a spread in temporal explanations could be because of the 10 different variable length utterances used in the ASVspoof 2017 dataset. Further, inspecting the T9 segment of several bonafide files in the training subset does not immediately reveal what M_2 is detecting; however, we noticed that many files have nonspeech (or silence) frames at their beginning. Therefore, M_2 may be using both the speech and nonspeech information across different temporal locations. Looking at the spectral explanations (right hand side of Table 4.18) we find that M_2 gives importance to the information present across all frequencies (0-8 kHz). To validate these findings, we repeat the above process across all the bonafide instances in the development and evaluation sets. As shown in Table 4.18 (third and fifth rows), we observe a consistency in the bonafide class explanations.

As for the spoofed class, in the training set, M_2 correctly classifies 1504 out of 1507 spoofed instances with more than 70% confidence. Table 4.18 (second row) shows the temporal and spectral explanations on these 1504 instances. The

Table 4.18: Distribution of temporal and spectral explanations generated from SLIME on the ASVspoof 2017 version 2.0 dataset. We take all the instances classified correctly with more than 70% confidence. **T**, **D** and **E** denote the training, development and evaluation subsets. **B**,**S** denote the bonafide and spoofed classes. T1-T10 and F1-F10 has the same meaning as in Tables 4.16 and 4.17. The numbers represent the count statistics for the top_1 explanations.

		What temporal information is the most critical?										*What spectral information is the most critical?*									
Set	**Class**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
T	**B**	13	12	60	215	0	9	2	132	**1047**	15	191	103	77	119	93	196	162	172	**285**	93
	S	**1208**	0	0	0	0	0	0	0	0	296	224	260	**410**	208	141	163	50	45	1	2
D	**B**	3	0	8	86	0	8	0	50	**521**	2	93	51	33	44	34	77	91	71	**122**	53
	S	**188**	0	0	0	0	0	0	0	0	167	60	44	**79**	30	23	44	21	37	15	2
E	**B**	49	13	13	116	0	9	0	98	**937**	30	100	93	96	124	82	126	170	190	**216**	52
	S	**1489**	0	0	0	0	0	0	0	0	**1668**	570	**647**	585	329	243	417	166	103	77	20

Table 4.19: Spectral and temporal explanations for all the **misclassified** bonafide (**B**) and spoofed (**S**) instances with more than 50% confidence in the development (**D**) and evaluation (**E**) sets. F1-F8 has the same meaning as in Table 4.17 and T1-T10 as in Table 4.16.

		What temporal information is the most critical?										*What spectral information is the most critical?*									
Class	**Set**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
B	**D**	4	0	0	0	0	0	0	0	0	3	1	2	1	2	0	0	0	1	0	0
	E	0	0	0	0	0	0	0	0	0	2	0	1	1	0	0	0	0	0	0	0
S	**D**	0	0	5	18	0	7	1	14	**224**	0	3	4	3	7	10	16	14	19	**21**	13
	E	62	34	28	221	0	23	2	172	**1540**	6	150	123	93	140	167	196	213	235	**311**	206

temporal explanation statistics suggest that the discriminative cue for replay spoofing detection appears either in the first or the last 400 ms of the signal (T1 or T10 segments) and that M_2 is not influenced much by the information present in temporal regions T2-T9. On the spectral explanations, we observe that M_2 is looking at the information present across all the frequencies. To validate these observations we repeat the process across all the spoofed instances in the development and evaluation sets. As shown in the fourth and sixth rows of Table 4.18 we observe similar explanations.

Now the question is what cue is there in the first and the last 400 ms of these instances? We inspect 50 spoof instances drawn randomly from the training set and find that (1) the majority of instances do not have non-speech/silence in the first 400 ms (2) and, these instances have DTMF-like (dual-tone multi-frequency) tones with speech (29 out of 50) and without speech (7 out of 50) in the first 400 ms of the signal. The last 400 ms of the spoof instances have the same property found in (1) and (2). One reason for this could be due to raw samples copied for audio instances less than 4 seconds duration.

We now analyse why a bonafide test instance is misclassified as spoofed and vice-versa. We hypothesize that a test utterance is misclassified if it does not exhibit its own class attributes but shows the attributes of the competing class. First, we look at why M_2 misclassifies a bonafide instance. We take all the bonafide instances misclassified as spoofed with more than 50% confidence[17] in the development and evaluation sets and generate temporal and spectral explanations. We show the results in the first two rows of Table 4.19. We find that these bonafide instances do not have the genuine class attributes, rather show the attributes of a spoofed class (top_1 corresponds to either T1 or T10 only), which explains the reason for misclassification.

Finally, we look at the spoofed audio instances in both the development and evaluation sets that were successful in fooling M_2 with more than 50% confidence. We find 269 (out of 950) such instances in the development set and 2088 (out of 12008) spoofed examples in the evaluation set. We show the explanations obtained from SLIME in the last two rows of Table 4.19. We find that these spoofed audio instances do not show the attributes of the spoofed class but appear to exhibit bonafide class properties (top_1 explanations distributed across T1-T10).

[17]We chose confidence more than 50% as there are very few instances with more than 70% confidence in the development and evaluation sets.

4.5.4 Interventions to test the significance of explanations

The previous Subsection 4.5.3 explained what M_2 is using to discriminate between bonafide and spoof instances using the SLIME algorithm (see Subsection 2.9.2). This section now aims at testing the significance of such explanations through two intervention experiments. The first one attempts to break the system by manipulating correctly detected spoofed test examples. The second intervention on the other hand aims at protecting an ASV system by manipulating misclassified spoofed examples.

Table 4.20: Intervention I: breaking the system. Demonstrating the effect on two spoof instances each in the training, development and evaluation sets.

Subset	Genuine class probability %		
	Instance ID	Before	After
Train	T_1002189	0.21	0.80
	T_1001687	0.22	0.80
Dev	D_1000884	0.21	0.85
	D_1000889	0.25	0.83
Eval	E_1004999	0.18	0.8
	E_1008476	0.19	0.81

Intervention I - Break the system

Here, our primary goal is to break M_2 from an attacker's perspective. In other words, we aim to increase the false alarm rate by manipulating correctly classified spoof instances so that M_2 judges them as genuine. We randomly take two spoofed audio instances from each of the training, development and evaluation sets that have been classified correctly with more than 70% confidence and replace their first and last 400 ms (T1 and T10) by a T1[18] segment of the most confidently classified bonafide signal in the training set (T_1000780). We then submit them to M_2 to see if they can pass as bonafide. Table 4.20 shows that M_2 now misclassifies them with high confidence. When we repeat this procedure for all the correctly detected spoofed instances in the development and evaluation sets, we observe a dramatic increase in the EER from 7.6% to 34.1% and from 10.6% to 29.7% respectively (first and second rows of Table 4.22).

[18]The main motivation here is to ensure that T1 and T10 (of the spoof instances) would not have any spoofed class attributes (as identified in Subsection 4.5.3) after the intervention. The best option was to pick a bonafide audio whose first 400 ms would contain mostly nonspeech/silence. That is why we pick T1 of T_1000780 (confident bonafide instance) which satisfies the criteria.

Intervention II - Protect the system

Here, our goal is to protect M_2 from a researchers' perspective (or an ASV system administrator). In other words, we aim to reduce the EER by manipulating misclassified spoofed instances so that M_2 judges them correctly. Since the training subset does not have any misclassified spoofed instances (EER is 0.0%) we randomly take two spoofed instances each from the development and evaluation sets that were successful in fooling M_2. From Subsection 4.5.3, we know that M_2 detects a spoofed signal correctly if spoofed attribute (DTMF tone and/or speech) appears in the first or last 400 ms (i.e T1 or T10 segment), we hypothesize that T1 and/or T10 of these four spoofed instances do not have such attributes. We generate temporal explanations for these four instances and find that the top_1 explanation does not favor T1 or T10.

Table 4.21: Demonstrating the effect of intervention II: protecting the system. Training subset has no misclassified instances.

Subset	Genuine class probability %		
	Instance ID	Before	After
Dev	D_1001544	0.81	0.23
	D_1000803	0.78	0.4
Eval	E_1003144	0.83	0.25
	E_1001926	0.82	0.29

Now, we remove raw samples from the beginning of these four instances to ensure that the speech signal occurs within the first 400 milliseconds. We choose the amount of samples to remove based on the original duration of the audio signal. For example, if the duration is between three to four seconds we remove the first 1200 ms samples. We then submit them to M_2 to see if they can be now detected as spoofed. Table 4.21 shows that M_2 now classifies them correctly as spoofed with high confidence. When we repeat this process across all the misclassified spoofed instances in the development and evaluation sets, we observe a reduction in the EER from 7.6% to 5.9% and from 10.6% to 7.8% respectively (first and third rows of Table 4.22).

Though intervention II did not completely reduce the EER of M_2 to 0% on the development and evaluation sets, it shows the potential of our analysis, and demonstrates how knowledge gained from such model explanations can help improve the detection performance. Upon closer analysis, we find that out of 269 misclassified spoof instances we intervened in the development subset, M_2 detects only 8 instances as spoofed with high confidence while a large number of instances were detected spoofed with low confidence. We find a similar observation on the evaluation set. Out of 2088 misclassified spoofed instances, only 88 instances were detected as spoofed with high confidence. This explains the

Table 4.22: EER% before and after the two interventions on M2.

Intervention	Dev	Eval
Initial EER	7.6	10.6
I: Break the system	34.13	29.76
II: Protect the system	5.9	7.8

reason for a small change in the EER.

4.5.5 Discussion

This section described the implementation and analysis of a replicated version of the LCNN model for replay spoofing detection using the ASVspoof 2017 dataset. Although our model M_1 trained on v1.0 of the dataset could not achieve 7% EER (as reported in [Lavrentyeva et al., 2017]), it achieved 9% EER on the evaluation set. Since version 1.0 of the dataset is obsolete, we focussed our analysis on version 2.0 of the dataset. Our adapted LCNN model M_2 reported an EER of about 10% on the evaluation set of this dataset. We used the SLIME algorithm to generate class explanations from both spectral and temporal perspectives. Our analysis shows that M_2 uses the first few milliseconds of the audio signal to make class predictions. We further demonstrated the significance of our analysis and findings by preprocessing the test signals which led to a predictable change in the EER on both the development and evaluation sets. Though these systems, including the state-of-the-art LCNN [Lavrentyeva et al., 2017], seem to be successful in discriminating between bonafide and spoofed speech, our analysis shows that to some extent they could be exploiting cues from the dataset which are unrelated to the problem. This raises a question about the integrity and trustworthiness of such systems. Further, the variability of patterns of signals (presence and absence of nonspeech frames in the beginning) within a class makes the problem difficult on this dataset. For example not all spoofed instances have a speech onset in the first few milliseconds of the audio signal and not all bonafide instances have nonspeech signals in the start. Our analysis showed how spoofing detection performance is correlated to the first few samples of the audio signals. This suggests that verifying dataset artefacts and potentially removing them prior to model training is important. Doing so will help models only exploit factors of interest relevant for replay spoofing detection (such as acoustic environment, playback device, recording device properties etc.)

4.6 A deeper look at the ASVspoof 2017 dataset

The ASVspoof challenge series (Subsection 2.3) has motivated research in protecting speech biometric systems against different variety of access attacks. The 2017 edition focused on replay spoofing attacks, and involved participants building and training systems on a provided dataset (ASVspoof 2017). More than 60 research papers have so far been published with this dataset, but none have sought to answer how successful countermeasures are in detecting spoofing attacks. This section shows how artefacts inherent to the dataset may be contributing to the apparent success of published systems.

4.6.1 Introduction

Most of the works on the ASVspoof 2017 dataset, as described in Section 2.4, aim at building countermeasure (CM) models with a focus on improving detection performance. Little attention is given to understand what these systems are learning to make predictions and what attributes are being exploited by the learning algorithms from the training data. Research often only focuses on improving scores based on a particular evaluation metric, with the equal error rate (EER) in this case. But showing better performance does not necessarily mean a system is trustworthy [Rusak et al., 2020, Hernandez-Orallo, 2019]. The EER is a scalar that does not provide any insights into what the model is learning to make a prediction.

Machine learning (ML) models learn to make decisions discovering patterns in the training data [Bishop, 2006]. Such models may easily exploit irrelevant cues, artefacts or confounders (if present) during the training optimisation. Unless explicitly accounted for (during training and inference) they often contribute to achieving good results and overestimate the actual performance that would be achieved in the wild on a test set [Kaufman et al., 2011]. As explained in Subsection 2.9.1, such artefacts can influence ML models across a variety of tasks including medicine, computer vision, music information retrieval, to name a few. Their trustworthiness is therefore called into question and some can behave much like a "horse" in machine learning [Hernandez-Orallo, 2019, Sturm, 2014], i.e. a model that provides excellent results using cues not relevant to the actual problem [Rodríguez-Algarra et al., 2019]. As highlighted in [Rosset et al., 2010], such biases can occur as a result of data collection, compilation, aggregation and partition. Such biases can have a severe impact on the trustworthiness of ML applications, and for domains such as finance, medicine and security (including ASV anti-spoofing) this can be catastrophic. Therefore, it is beneficial to perform an in-depth dataset analysis [Tommasi et al., 2015], to detect the presence of artefacts or confounders [Stowell et al., 2019], ensur-

ing models do not exploit irrelevant factors during training, and therefore yield reliable performance estimates.

The work in this section builds upon the findings from Section 4.5, which suggests that v2.0 of the ASVspoof 2017 dataset contains some recording artefacts that bias the performance estimates. It was shown that the CNN highly attends to the first few milliseconds to make decisions. Further analysing a few confidently classified test audio signals showed the presence of dual tone multi-frequency (DTMF) sounds (or speech signals) within the first few milliseconds for spoofed audio but nonspeech or silence in case of the bonafide audio signals. Therefore we hypothesize that the training and development subsets of the same dataset might contain such DTMF sounds and other confounders/artefacts that might influence model decisions. Understanding their statistics enables the building of trustworthy CMs using this dataset. To the best of our knowledge there has been no other works in understanding this dataset, its artefacts and its impact on machine learning CM models.

To this end, the next Subsection 4.6.2 first provides a description of different countermeasure models investigated to understand the influence of dataset artefacts. As for the artefact details, we use the results of qualitative analysis on v2.0 of this dataset from Subsection 3.2.3. Then the following Subsection 4.6.3 performs an in-depth analysis through intervention experiments to uncover the dependencies of the identified artefacts on countermeasure model decisions. Furthermore, Subsection 4.6.4 performs additional intervention experiments to confirm the findings on the influence of artefacts. It demonstrates how knowledge of cues/artefacts in the training data can be used to compromise model decisions. Finally, Subsection 4.6.5 provides a summary of work done in this section.

4.6.2 Experimental design and evaluation

This section describes different features, classifiers and performance metrics considered to evaluate the influence of the artefacts (Subsection 3.2.3) on CM models. The section also discusses the initial performance of these models.

Features

We use CQCCs, i-vectors and power spectrograms as input features. The main reasons for choosing them are: CQCCs and its variants have been studied extensively on the ASVspoof 2017 dataset [Delgado et al., 2018, Witkowski et al., 2017, Yang et al., 2018a]; CQCC-based i-vectors have been used as a baseline system [Delgado et al., 2018]; power spectrogram features have shown the best performance during the ASVspoof 2017 challenge [Lavrentyeva et al., 2017].

Section 2.5 provides background on these features.

Classifiers

We now describe five different backend models: GMMs, Cosine Distance, SVMs and two CNNs considered in this study. The motivation to chose them is twofold. First, the above mentioned classifiers have been widely used for spoofing detection tasks [Delgado et al., 2018, Lavrentyeva et al., 2017, Sahidullah et al., 2015]. Second, we aim to demonstrate that dataset artefacts can affect any ML model and that the issues discussed do not revolve around a specific model.

GMM: We train one GMM each for the bonafide and spoof classes using 512 mixture components with random initialisation. We use 60 dimensional CQCC features extracted using the setup from [Delgado et al., 2018] to train the GMMs. During testing, for each test utterance X (with T feature vectors) a score is obtained using the log-likelihood ratio between the bonafide and spoofed GMM as described in Eq. (4.1).

Cosine: We compute 100-dimensional i-vectors using the same 60 dimensional CQCC features used in GMMs for the entire dataset. We compute the mean i-vector corresponding to the bonafide and spoof classes in the training set and use them as the representative models. During testing, a similarity score is computed between a test i-vector and the model i-vector using the cosine distance metric:

$$\cos(\theta) = \frac{\mathbf{X} \cdot \mathbf{Y}}{||\mathbf{X}||||\mathbf{Y}||} \tag{4.2}$$

where $\mathbf{X}$ represents the test i-vector and $\mathbf{Y}$ the model i-vector. The final score is then obtained by taking the difference between the bonafide and spoof model scores. We follow the same i-vector setup from [Delgado et al., 2018]. It is also worth noting that this model has far fewer parameters than the others.

SVM: We train a binary SVM classifier with a linear kernel using mean-variance normalised i-vector features, with mean-variance values computed on the training set. Here, we use the same 100 dimensional CQCC-based i-vectors used in the Cosine model. We use the Scikit-learn [Pedregosa et al., 2011] library with default parameters for training the SVM model.

CNN: We use two different CNN models, CNN_1 and CNN_2. We take CNN_1 as a pretrained model from Section 4.5.2 (model M_2). CNN_2 is a newly proposed architecture with only $36,174$ free parameters and comprises 3 convolutional (Conv) and 2 fully connected (FC) layers unlike CNN_1 which is very deep com-

prising 9 convolutional layers. Both CNNs operate on a fixed input representation and use a batch normalisation layer before applying ReLU nonlinearity in every Conv and FC layers. The input representation, model training and testing approach used in CNN_2 is same as in CNN_1 with the key difference being the input duration and parameters used in spectrogram computation. CNN_2 operates on 3 seconds input (CNN_1 uses 4 seconds duration) computed using a 512-point FFT, 32 ms frame window and 10 ms hop size. Additional details on the CNN architectures are provided in Appendix A.

Table 4.23: Initial model performance. Θ = EER decision threshold.

Model	Set	Θ	TP	FN	FP	TN	EER%
CNN_1	Dev	0.6663	701	59	74	876	7.7
	Eval	0.7467	1159	139	1286	10722	10.7
CNN_2	Dev	0.6	704	56	70	880	7.37
	Eval	0.842	1124	174	1609	10399	13.4
Cosine	Dev	0.125	679	81	101	849	10.6
	Eval	0.181	1105	193	1779	10228	14.8
SVM	Dev	0.3972	678	82	103	847	10.8
	Eval	0.506	1094	204	1883	10125	15.6
GMM	Dev	0.3334	690	70	87	863	9.2
	Eval	0.7120	1119	179	1656	10352	13.7

Performance measures

We use the EER metric to evaluate the performance of CM models. In order to derive more insights in understanding the impact of the artefacts on this dataset, we further report true positive (TP), false negative (FN), false positive (FP), true negative (TN), false acceptance rate (FAR) and false rejection rate (FRR) for the bonafide class. Subsection 3.5.1 provides a description of these metrics.

Preliminary results

We train all our CM models using the training set and validate them on the development set. Table 4.23 summarises our preliminary results. Our CM models Cosine and GMM show consistent performance as reported in [Delgado et al., 2018]. Although CNN_2 and CNN_1 show similar performance on the development set, CNN_2 performs poorly on the evaluation set. A possible reason could be due to the simple 4 hidden layer architecture used by CNN_2 in comparison to the 10 hidden layer representation in CNN_1. However, CNN_2 outperforms both Cosine, GMM and SVM on both the development and evaluation subsets. We also record the individual EER threshold for each model (for each test set)

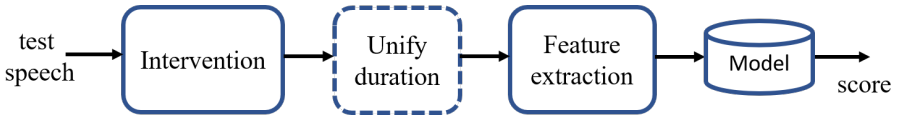


Figure 4.5: Block diagram: intervention pipeline towards understanding the influence of artefacts on the predictions of countermeasure models.

that we use in the next section to understand the influence of the dataset cues highlighted in Subsection 3.2.3 on model predictions.

4.6.3 Understanding the influence of dataset artefacts

In this section we thoroughly study the impact of artefacts explained in Subsection 3.2.3 on the CM model decisions. More precisely we focus on understanding the influence of pattern difference, BCS and DTMF sounds on model prediction through intervention experiments illustrated in Fig. 4.5. We call this setup as *inference-time intervention* because we do not retrain any of our CM models, rather use our pretrained CMs. Here the intervention module updates the test signal by exploiting information about the dataset artefacts which we pass as a side information. Features are then computed on the updated test signal and scoring is performed using a pretrained model. The optional *unify-duration* module truncates or replicates audio samples to create a fixed-duration input representation. This is applicable only for the CNNs.

Finally, we demonstrate a use-case where an attacker despite having physical access to such cues (for example, a BCS) is still capable of manipulating CM decisions by crafting synthetic artefacts. With these interventions we confirm that both frame-level and utterance-level CM models can exploit the artefacts in this dataset raising concerns about their trustworthiness and reliability of the published results.

Impact of "BCS" on prediction

As described in Subsection 3.2.3, the training set contains a large proportion (36.36%) of bonafide examples with BCS artefacts in comparison to only 2.45% of spoof examples. BCS, if present in an audio recording, usually occurs within a 100 ms time window and is found either at the start or at the end. Although few (10.81%) bonafide class audio files in the training set have BCS at the end, our preliminary interventions showed they have no impact on model predictions. However, we find a substantial influence for BCS found at the start of the audio recordings. We do not perform this intervention on the spoof class as the BCS serves as a cue for the bonafide class. Therefore we hypothesize and demonstrate that BCS serves as one kind of bonafide signature on this dataset. And if this

Table 4.24: BCS intervention results. TFI: test files intervened, which corresponds to TP cases (Table 4.23) identified to contain BCS artefact. Prop: proportion of files that changed class label.

Model	Set	# TFI	FN	Prop (%)
CNN_1	Dev	177	+34	19.21
	Eval	513	+118	23.0
CNN_2	Dev	175	+12	6.85
	Eval	508	+60	11.81
Cosine	Dev	159	+8	5.03
	Eval	486	+32	6.58
SVM	Dev	159	+6	3.77
	Eval	491	+32	6.51
GMM	Dev	173	+13	7.51
	Eval	508	+56	11.02

signature is not removed, machine learning CMs can easily exploit it. To this end, we take all the TPs for the bonafide class that contain a BCS at the start and run this intervention on them.

Here our intervention module (Fig. 4.5) takes a BCS annotation file containing a list of files having a BCS as side information. It then discards the first 100 ms audio samples from the test utterance before extracting features and obtaining a classification score. Table 4.24 summarises the results. As expected, dropping BCS samples causes models to misclassify bonafide test signals raising the false negatives. Interestingly, we find CNN_1 to be more sensitive in contrast to CNN_2 and other models. A possible explanation is since CNN_1 uses a 4 second representation in contrast to the 3 seconds one for CNN_2. The above mentioned representation of CNN_1 contains more replicated copies of shorter audio segments, which propagates artefacts (see Fig. 4.6).

Impact of "DTMF" on prediction

During replay data collection a number of bonafide utterances were first concatenated using a DTMF sound to mark the utterance boundary and replay attacks were simulated on them. The individual replayed utterance was then retrieved based on this marker [Kinnunen et al., 2017b]. As outlined in Subsection 3.2.3, some spoof audio files in the training set have DTMF sounds (low or loud) which are not present in the bonafide files. Do DTMF sounds provide cues for the spoof class? Do these dataset artefacts bias our ML models? We perform interventions to understand this. As the ground truth of DTMF artefacts for the spoof class in the evaluation set is unavailable the present study does not include this intervention on them. To this end, we take all the TNs for the spoof class in the development set that contain a DTMF artefact and run

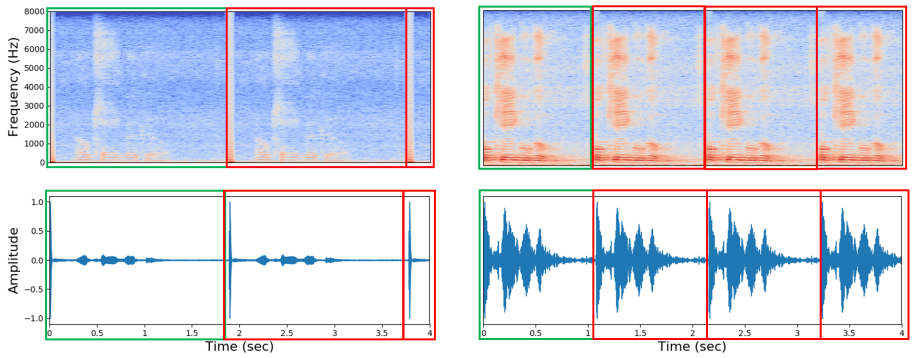


Figure 4.6: Spectrogram (top) and raw audio (bottom) of "Ok google". Left represents a bonafide example and right its replayed version. The green rectangular box highlights the original audio and the red box shows signal replication to create a fixed duration (4 seconds) input representation. Two use cases are reflected: pattern difference and BCS. It shows how artefacts spread while creating an input representation with fixed duration.

this intervention.

We pass the file identifier containing a DTMF as side information to the intervention module (Fig. 4.5) which removes the first 250 ms audio samples from them before extracting features and obtaining a classification score. Table 4.25 summarises the intervention results. We see a negligible proportion of intervened files affected from this intervention, which suggests that DTMF sounds do not provide a substantial bias on CM decisions. Another interpretation is the fact that the spoof signals acquire other channel characteristics during the replay simulation. Therefore, their impact may be negligible when audio signals are replayed in noisy acoustic conditions.

Table 4.25: DTMF intervention results for the development set spoof files identified to contain a DTMF. Prop has the same meaning as in Table 4.24. TFI: test files intervened.

Model	# TFI	FP	Prop (%)
CNN_1	136	+3	2.2
CNN_2	144	+3	2.08
Cosine	145	0	0.0
SVM	145	0	0.0
GMM	141	+1	0.71

Impact of "pattern difference" on prediction

The previous two experiments involved removing BCS and DTMF artefacts from the test files identified to contain such artefacts. In this experiment we

remove audio samples before and after the actual speech utterance during testing ensuring that both bonafide and spoofed audio recordings now have similar audio patterns. This also means that BCS or DTMF gets removed, if present, in this intervention experiment. Thus, BCS and DTMF experiments can be thought of as a special case of pattern difference interventions but performed on a small set of test files identified to contain such artefacts.

Here, the intervention module uses speech endpoints as side information and discards audio samples before and after the actual speech utterance. We use our manual speech endpoint annotations that we prepared during dataset inspection (Subsection 3.2.3). This intervention ensures that both bonafide and spoof test recordings now have similar audio patterns. Following our prior findings from Subsection 4.5.3 we hypothesize that the pattern difference favours the bonafide class. To confirm this, we take all the TPs for the bonafide class and all the FPs for the spoof class (from Table 4.23) and run this intervention on them. Furthermore, due to the large numbers of spoof files in the evaluation set we could not perform manual inspection on them. Therefore, we are unable to run this intervention on them in the present study. We evaluate all our countermeasure model performance using the original decision thresholds from Table 4.23.

Table 4.26: Pattern difference intervention results. To be compared with Table 4.23. '+', '-' denotes an absolute increase/decrease. Ground truth annotations for spoof files in the evaluation set are unavailable (indicated by −).

Intervention on		Bonafide class		Spoof class	
Model	Set	FN	FRR %	FP	FAR %
CNN_1	Dev	+334	+43.95	-49	-5.16
	Eval	+519	+39.98	−	−
CNN_2	Dev	+73	+9.61	-35	-3.68
	Eval	+289	+22.27	−	−
Cosine	Dev	+155	+20.39	-53	-5.58
	Eval	+352	+27.12	−	−
SVM	Dev	+174	+22.89	-52	-5.47
	Eval	+349	+26.89	−	−
GMM	Dev	+170	+22.37	-41	-4.32
	Eval	+429	+33.13	−	−

Table 4.26 summarises the results of this intervention. As expected, a large number of bonafide test examples on both the development and evaluation sets are now misclassified by all our ML models as shown from the increased FN and FRR% metrics. On the spoof class (development set) we find a drop in the FP and FAR% metrics for all ML models. These results confirm our hypothesis about this pattern difference on this dataset. It indeed favours the bonafide

class. This makes sense since a large proportion of bonafide audio files in the training set have silence/nonspeech in the first 300 ms while the spoof class contains speech.

4.6.4 Manipulating model decisions

Now we aim to manipulate countermeasure decisions by using the knowledge acquired so far through our intervention experiments. Among different artefacts, we find that "pattern" difference and "BCS" artefacts favour the bonafide class as evident from the increase in FRR% when we removed them from the bonafide test files. Therefore, using the BCS artefact as a bonafide class signature we perform interventions on all the FNs (misclassified bonafide files) and FPs (correctly detected spoof files) from Table 4.23. From these interventions we demonstrate that machine learning CM models trained on this dataset can indeed be manipulated using this cue.

Here the intervention module (Fig. 4.5) takes as side-information a "BCS" signature and appends it to the start of test audio recordings before passing on to the other modules for feature extraction and scoring. As a BCS signature we use the first 100 ms samples of the most confidently classified bonafide audio "T_1001039.wav" containing a BCS artefact in the training set. It should be noted that we did a similar line of study in Sections 4.2 and 4.5 but this study is different in terms of the signature we used for interventions. In Section 4.2 we used 60 ms zero-valued silence as a signature to fool GMM-based countermeasures on version 1.0 of this dataset. In Section 4.5, we find that CNNs give strong emphasis on the first 400 ms for class discrimination. And, we used the initial 400 ms samples[19] as a signature to fool the prediction of the CNN countermeasure.

Table 4.27 shows the intervention results in terms of absolute increase/decrease in the error metrics. The consistency in the drop of FN and FRR% and the increase in FP and FAR% across all ML models confirm our hypothesis about BCS. It indeed serves as a strong cue that a model attends to form bonafide class predictions. The GMM and CNNs in particular show a high impact of this intervention on the evaluation set. For example 124 misclassified genuine files are now correctly classified by CNN_1 (out of 139) and 142 by the GMM (out of 179). Furthermore FAR raises by more than 40% for the GMM and CNNs demonstrating that a large amount of correctly detected spoof files are now able to bypass them. Even though i-vectors are computed on CQCC features the impact on Cosine and SVM models that operate on i-vectors is much smaller than GMMs. A possible reason for this is that i-vectors involve feature aggre-

[19]Taken from the most confidently classified bonafide audio signal in the training set.

Table 4.27: Manipulating model decisions using BCS. To be compared with Table 4.23. '+', '-' denotes an absolute increase/decrease.

Intervention on		Misclassified bonafide files		Correctly detected spoof files	
Model	Set	FN	FRR %	FP	FAR %
CNN_1	Dev	-46	-6.05	+446	+46.95
	Eval	-129	-9.94	+4909	+40.88
CNN_2	Dev	-56	-7.37	+479	+50.42
	Eval	-153	-11.79	+4937	+41.11
Cosine	Dev	-37	-4.87	+130	+13.68
	Eval	-54	-4.16	+1857	+15.46
SVM	Dev	-33	-4.34	+106	+11.16
	Eval	-53	-4.08	+1952	+16.26
GMM	Dev	-51	-6.71	+325	+34.21
	Eval	-142	-10.94	+5732	+47.73

gation across all time frames during super-vector computation which may have reduced the influence of BCS on the final i-vector feature. Furthermore, Fig. 4.7 provides additional insights through score visualisations illustrating how confidently classified spoof test examples (true negatives) are now misclassified by our CMs.

An interesting question we now ask is: What if an attacker did not have access to this BCS signature? Can they still fool countermeasure models trained on this dataset using a synthesized burst sound? To demonstrate this, we now repeat the same BCS interventions but use 100 ms white noise as a signature to fool ML models. We experiment with synthetic noise at different signal to noise ratios (SNR) and demonstrate that white noise with enough power can fool ML decisions serving as a cue for the bonafide class. To ensure that the power of the original and manipulated speech signal is equivalent after adding noise, we first normalise the noise samples. Let X_i represent a test audio signal and n_i be the noise samples drawn from a standard normal distribution. Therefore the signal to noise ratio (SNR) can be written as:

$$\text{SNR} = \log_{10}\left[\frac{\text{Var}(X_i)}{\text{Var}(\alpha \times n_i)}\right] \tag{4.3}$$

where α is the scalar we want to compute for a given X_i and an SNR and Var(.) represents variance. Thus:

$$\alpha = \sqrt{\text{Var}(X_i) \times 10^{-\text{SNR}}}. \tag{4.4}$$

Using α in Equation 4.4, the intervention module (Fig. 4.5) normalises the random noise n_i before appending it to X_i. The updated signal is then prop-

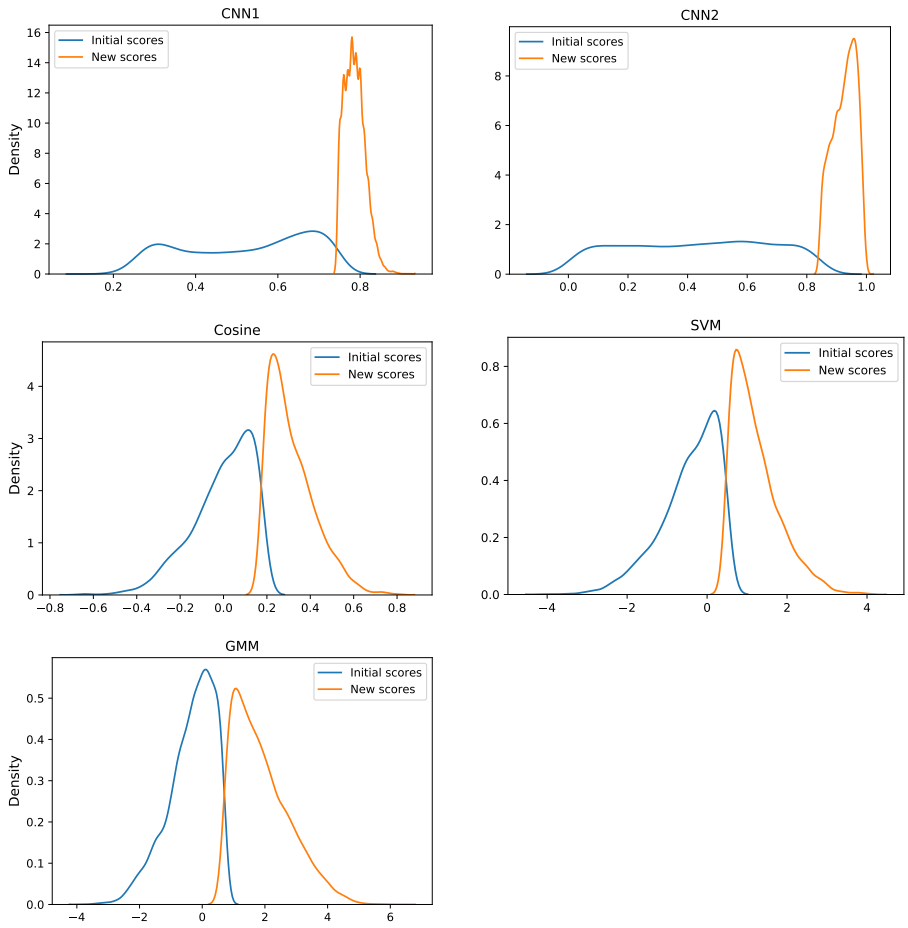


Figure 4.7: Score distribution (before and after) illustrating how true negatives (spoof files) in the evaluation set are misclassified after adding a BCS signature on them. X-axis represents the countermeasure scores.

agated for feature extraction and scoring. We investigate the impact of noise at different SNR levels and at the start and random time locations. Due to similarity in the trend for different SNRs, we report the findings only for 0 and 6 dB SNR noise.

Tables 4.28 and 4.29 summarise the results of performing this intervention adding white noise at random time locations and at the start of test signals. In general, irrespective of locations where we inject this signature, on both the development and evaluation sets we find the impact to become less effective as we increase the SNR. Although we see a similar trend (as in Table 4.27) in FRR% and FAR% for all our CMs, we find much smaller impact for the GMM using noise in comparison to BCS (compare Table 4.27 and 4.28). A possible reason could be that we normalise the noise with respect to the original signal

Table 4.28: Manipulating model decisions injecting white noise at random time locations of the intervened test signal. To be compared with Table 4.23.

Intervention on			Misclassified bonafide trials		Correctly detected spoof trials	
Model	SNR	Set	FN	FRR	FP	FAR
CNN_1	0	Dev	-26	-3.42	+394	+41.47
		Eval	-114	-8.78	+4583	+38.17
	6	Dev	-38	-5.0	+176	+18.53
		Eval	-94	-7.24	+1138	+9.48
CNN_2	0	Dev	-43	-5.66	+662	+69.68
		Eval	-136	-10.48	+6748	+56.2
	6	Dev	-40	-5.26	+164	+17.26
		Eval	-73	-5.62	+2113	+17.6
Cosine	0	Dev	-63	-8.29	+161	+16.95
		Eval	-78	-6.01	+1425	+11.87
	6	Dev	-13	-1.71	+14	+1.47
		Eval	-13	-1.0	+240	+2.01
SVM	0	Dev	-61	-8.03	+139	+14.63
		Eval	-74	-5.70	+1333	+11.10
	6	Dev	-11	-1.45	+20	+2.11
		Eval	-19	-1.46	+267	+2.22
GMM	0	Dev	-17	-2.24	+46	+4.84
		Eval	-20	-1.54	+660	+5.50
	6	Dev	-27	-3.55	+49	+5.16
		Eval	-23	-1.77	+368	+3.06

power (Eq. 4.4) which is not performed with BCS. We simply copy the BCS (raw samples containing BCS) and append it to the test signal during the intervention (Fig. 4.5).

Fooling CM decisions using "silence". A final set of intervention experiments we perform is towards evaluating CM robustness against silence (zero-valued samples). It was already demonstrated on version 1.0 of this dataset how strong impact silence had on class decisions (see Subsection 4.2.3). During our dataset inspection on version 2.0 of this dataset, we found some of the bonafide audio files in the training set still contain silence of more than 10 ms in the start (see Section 3.2.3) which is not found in the spoof class. As highlighted in Section 3.2.3 the training set has 288 (out of 1507) bonafide files containing silence of more than 10 ms. Out of these files, 68 files have more than 70 ms silence and 37 with more than 100 ms silence in the start. Therefore, we hypothesize that CMs trained on the version 2.0 of this dataset may be still exploiting silence as one potential cue for the bonafide class. To this end, we repeat the same interventions as we did for BCS and noise, but now we use zero-valued samples

Table 4.29: Same as in Table 4.28 but we append white noise at the beginning now.

Intervention on			Misclassified bonafide trials		Correctly detected spoof trials	
Model	SNR	Set	FN	FRR	FP	FAR
CNN1	0	D	-32	-4.21	+457	+48.11
		E	-120	-9.24	+5875	+48.93
	6	D	-52	-6.84	+292	+30.74
		E	-106	-8.17	+2519	+20.98
CNN2	0	D	-52	-6.84	+798	+84.00
		E	-161	-12.4	+8537	+71.09
	6	D	-49	-6.45	+289	+30.42
		E	-118	-9.09	+3794	+31.6
Cosine	0	D	-56	-7.37	+107	+11.26
		E	-74	-5.70	+1239	+10.32
	6	D	-20	-2.63	+17	+1.79
		E	-21	-1.62	+291	+2.42
SVM	0	D	-59	-7.76	+101	+10.63
		E	-78	-6.01	+1201	+10.00
	6	D	-15	-1.97	+17	+1.79
		E	-20	-1.54	+288	+2.40
GMM	0	D	-8	-1.05	+17	+1.79
		E	-8	-0.62	+617	+5.14
	6	D	-30	-3.95	+30	+3.16
		E	-29	-2.23	+330	+2.75

as a bonafide class signature (at start and random time locations) with an aim to fool CM decisions. We perform this intervention using 10 ms and 100 ms duration to demonstrate how varying duration effects CM decisions.

Tables 4.30 and 4.31 summarise the results of these intervention experiments injecting silence at the start and at random time locations of test signals. Overall, both these tables illustrate a similar pattern. The impact becomes higher as we increase zero-valued samples from 10 ms to 100 ms. This holds true for both the development and evaluation sets. Interestingly, CNN_2 appears to be more sensitive than CNN_1. A possible interpretation to this corresponds to the framing size used during spectrogram computation. A frame size of more than 100 ms indicates that even though all 100 ms samples are replaced with silence, a frame may contain some speech/nonspeech information. To this end, CNN_1 uses a 108 ms frame size (1728 point FFT) while CNN_2 uses only 32 ms (512 point FFT) as frame size to compute spectrograms. Both use 10 ms frame shift. Thus, appending silence of 10 or 100 ms would impact CNN_2 more in comparison to CNN_1 due to occurrences of such silences (in the bonafide class) during model training. To derive further understanding on the impact of this

Table 4.30: Manipulating CM decisions injecting **silence** at the start of the intervened test signal. To be compared with Table 4.23

Intervention on			Misclassified bonafide trials		Correctly detected spoof trials	
Model	Silence	Set	FN	FRR	FP	FAR
CNN1	10 ms	D	0	0	+1	+0.11
		E	-2	-0.15	+33	+0.27
	100 ms	D	-9	-1.18	+24	+2.53
		E	-8	-0.62	+199	+1.66
CNN2	10 ms	D	-17	-2.24	+28	+2.95
		E	-17	-1.31	+219	+1.82
	100 ms	D	-51	-6.71	+711	+74.84
		E	-134	-10.32	+6226	+51.85
Cosine	10 ms	D	-16	-2.11	+4	+0.42
		E	-9	-0.69	+97	+0.81
	100 ms	D	-29	-3.82	+35	+3.68
		E	-37	-2.85	+479	+3.99
SVM	10 ms	D	-9	-1.18	+9	+0.95
		E	-12	-0.92	+12	+0.10
	100 ms	D	-33	-4.34	+33	+3.47
		E	-34	-2.62	+34	+0.28
GMM	10 ms	D	-7	-0.92	+6	+0.63
		E	-5	-0.39	+78	+0.65
	100 ms	D	-32	-4.21	+44	+4.63
		E	-44	-3.39	+479	+3.99

intervention we look at the score distribution for CNN_2. Fig. 4.8 shows this. Overall, these results indicate that silence does provide cues for the bonafide class on v2.0 of this dataset.

4.6.5 Discussion

The performance of any data-driven ML task highly depends on the training data fed to the learning algorithm. They learn to make decisions by exploiting the underlying pattern within the training data. Therefore, artefacts and confounders, if present in the dataset, can introduce biases in model decisions raising questions on their reliability and trustworthiness [Hernandez-Orallo, 2019, Sturm, 2016]. As explained in Subsection 2.9.1 such issues can affect a wide range of ML tasks, and the impact caused by these biases in domains such as finance, medicine and security (including ASV anti-spoofing) can be very costly. Therefore, it is important to ensure that dataset artefacts that introduce biases in ML decisions are taken into account to build reliable ML models.

In this direction, this section focussed on security for voice biometric using a benchmark ASVspoof 2017 dataset which is a popular dataset for replay

Table 4.31: Same as in Table 4.30 but now we inject **silence** at **random** time locations.

Intervention on			Misclassified bonafide trials		Correctly detected spoof trials	
Model	Silence	Set	FN	FRR	FP	FAR
CNN1	10 ms	D	0	0.0	0	0.0
		E	-2	-0.15	+34	+0.28
	100 ms	D	-11	-1.45	+53	+5.58
		E	-9	-0.69	+295	+2.46
CNN2	10 ms	D	-16	-2.11	+59	+6.21
		E	-34	-2.62	+355	+2.96
	100 ms	D	-52	-6.84	+814	+85.68
		E	-142	-10.94	+7310	+60.88
Cosine	10 ms	D	-10	-1.32	+9	+0.95
		E	-10	-0.77	+100	+0.83
	100 ms	D	-19	-2.5	+31	+3.26
		E	-35	-2.7	+534	+4.45
SVM	10 ms	D	-4	-0.53	+11	+1.16
		E	-9	-0.69	+102	+0.85
	100 ms	D	-26	-3.42	+33	+3.47
		E	-35	-2.7	+533	+4.44
GMM	10 ms	D	-3	-0.39	+8	+0.84
		E	-10	-0.77	+103	+0.86
	100 ms	D	-30	-3.95	+73	+7.68
		E	-26	-2.0	+655	+5.45

spoofing attack detection used in more than 60 published research papers. We identified and investigated the impact of artefacts (see Subsection 3.2.3) on this dataset that machine learning models exploit to form decisions. Among different artefacts, we found that burst click sounds (BCS) provide strong cues for the bonafide class. Interestingly, experimental results for interventions using DTMF sounds showed that they have no influence on model decisions (Subsection 4.6.3). We also find that silence (zero valued samples) still serves as a cue for the bonafide class on version 2.0 of this dataset. Initially, in Section 4.2 we had shown how silence influenced model decisions on version 1.0 of this dataset. Subsequently version 2.0 was released by the ASVspoof organisers [Delgado et al., 2018] fixing such issues. However, our intervention results suggest that this problem is not solved completely, and that CM models still exploit silence as one potential cue in class discrimination.

Among 10 different utterances (see Table 3.1) used in this dataset, S02 "Ok Google" is the shortest one with an average duration of about 0.7 to 0.8 seconds. However, the dataset contains a large number of S02 examples with more than 2 seconds duration indicating more than half of its contents being nonspeech,

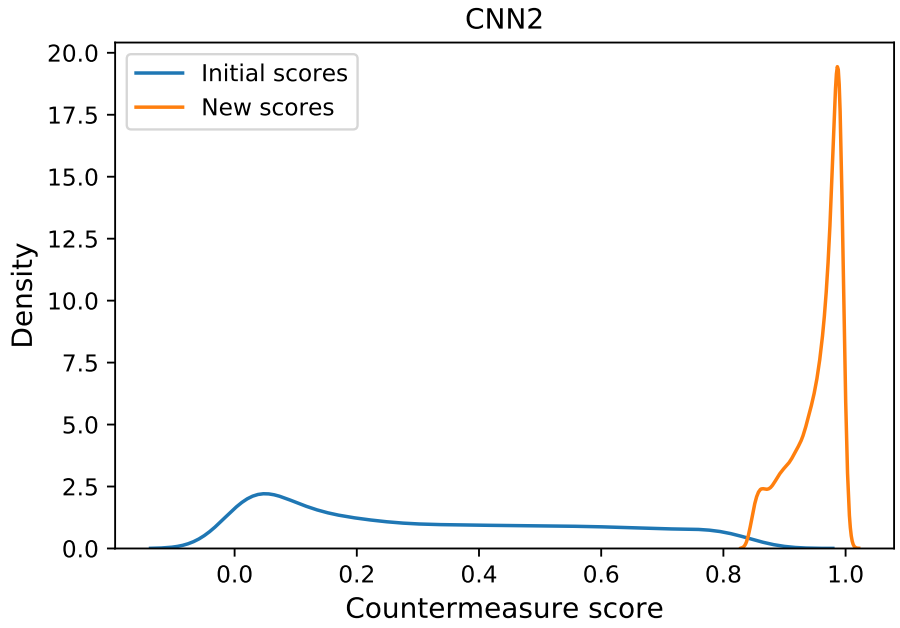


Figure 4.8: Score distributions of spoof files in the evaluation set that were originally detected correctly by CNN_2 and get misclassified adding 100 ms silence at random time locations.

noise or silence. As a result of this, a reliable detection of this sentence may be difficult and that the attacks we demonstrated using dataset cues may have more impact on S02 than the other nine phrases. To confirm this, we visualise the distribution of how the proportion (in terms of impact) looks across the ten phrases. Fig. 4.9 summarises this and confirms our hypothesis.

We find the work of [Fang et al., 2018b] to be closely relevant to ours. The authors study the effect of enhancing the quality of stolen speech using generative adversarial networks before performing a replay attack. They demonstrate a significant increase in the EER for both the baseline GMM and CNN countermeasures on the ASVspoof 2017 dataset. However, our current work demonstrates that knowledge of artefacts in a dataset can also be used to manipulate model predictions (Subsection 4.6.4). This is much simpler than training a GAN on a specific dataset. Our in-depth analysis of this dataset and experimental results from different interventions confirms the presence of a "horse" in machine learning [Hernandez-Orallo, 2019, Sturm, 2014] for anti-spoofing applied to ASVspoof 2017 dataset. Furthermore, none of the research results published in this dataset (more than 60 papers) have accounted for these artefacts, further indicating that the countermeasures showing impressive results may not be fully reliable and trustworthy as their decision process involves the contribution of

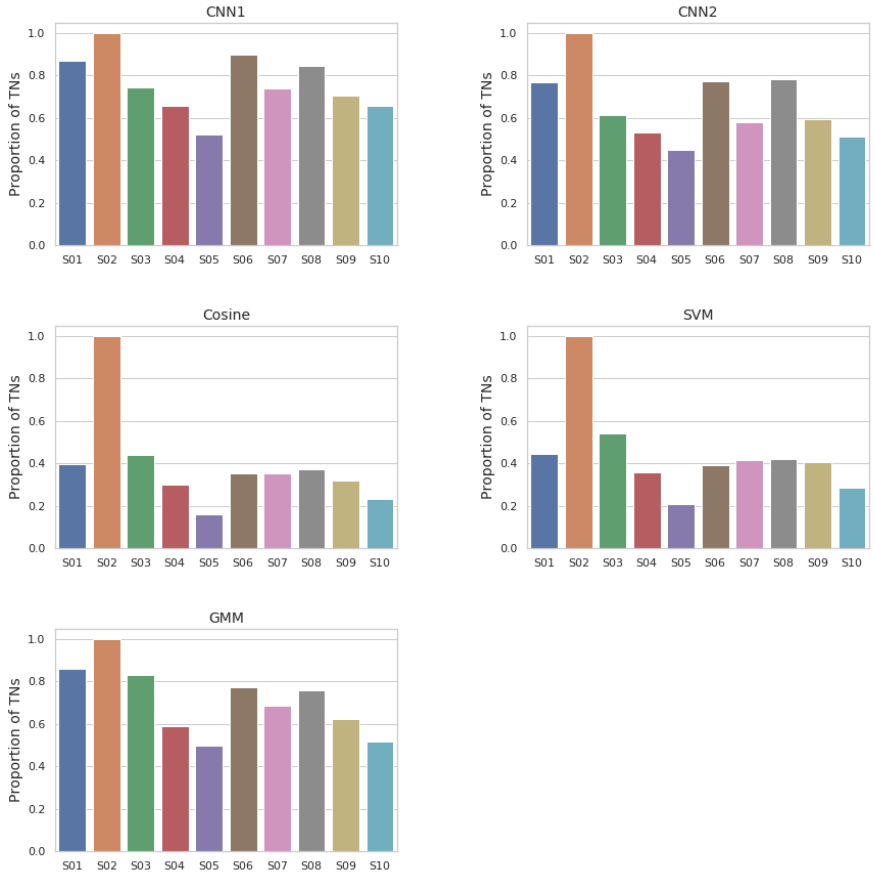


Figure 4.9: Impact of the BCS intervention on correctly detected spoof files across ten different phrases (S01 through S10 defined in Table 3.1) of the ASVspoof 2017 dataset. Shown results illustrate the proportion for different phrases for all our CM models.

these artefacts which are not related to the actual problem.

4.7 Summary

This chapter presented a detailed study towards understanding replay spoofing detection systems by investigating existing methodologies and techniques from the literature. Several interesting insights have been discovered revolving around the ASVspoof 2017 dataset, a benchmark spoofing dataset and state-of-the-art countermeasure models on this dataset. This section provides a summary of these findings.

In Section 4.2, our experimental results demonstrated that CM models exploited confounding factors in this dataset (version 1.0) that are not relevant to

the replay detection problem, overestimating the actual performance. Furthermore, the poor performance using hand-crafted features indicated that crafting features to capture unknown attack conditions might be difficult, suggesting data-driven models could be a way for exploration. To this end, Section 4.3 attempted to replicate the best performing deep model (LCNN) of the ASVspoof 2017 challenge. It was found that replicating this model using only the published details was not possible. Despite trying alternative network architectures it was found that achieving good test generalisation was a major challenge on this dataset.

Why is spoofing detection on this dataset difficult? Section 4.4 attempted to understand this. It was found that due to inherent dataset issues, making confident conclusions whether reverberation noise or some device-specific attributes provide a cue to replay signal discrimination on this dataset is difficult. Extending this, the next Section 4.5 developed a CNN model that showed comparable performance to LCNN, and used it to understand what it has learned to detect spoofing using a method from interpretable machine learning. Results demonstrated that the model was paying attention to the first few milliseconds of each input recording to make class decisions. Motivated by this, Section 4.6 performed an in-depth study on the ASVspoof 2017 v2.0 dataset, discovered artefacts/confounders and demonstrated that CM models benefit from exploiting them in their decision making.

The work reported in this chapter has been published (and is under review) in peer-reviewed international conferences and journals.

Chapter 5

Design of novel spoofing countermeasures

5.1 Introduction

This chapter proposes novel methods for the design of countermeasures for replay spoofing attacks while also focusing on model robustness and avoiding biases in the datasets. The work reported in this chapter uses both the ASVspoof 2017 v2.0 and ASVspoof 2019 PA datasets except for Section 5.3 which mainly focuses on the ASVspoof 2017 v2.0 dataset. Section 5.2 describes our proposed ensemble model comprising several shallow and deep models reporting promising results on the benchmark ASVspoof 2019 PA dataset. It also discusses how countermeasure models show biased performance with zero-valued silences in the PA dataset, and subsequently proposes simple preprocessing methods to overcome them. This work is a result of our participation in the recent ASVspoof 2019 challenge, and was published in [Chettri et al., 2019]. Then Section 5.3 extends work from Section 4.6 and proposes methods to mitigate the impact of dataset biases and help countermeasure models become more robust against manipulations using dataset specific cues (see Section 4.6). The next Section 5.4 analyses how spoofing detection performance varies across different subbands using CNNs trained on spectrograms. It then proposes a joint subband modeling framework which outperforms models trained on fullband spectrograms. Furthermore, this section performs a cross-dataset evaluation of these models on the ASVspoof 2019 real PA test set (described in Subsection 3.3.3). Motivated from the widespread popularity of the GMM backend classifier (which is a generative model) in spoofing detection, Section 5.5 proposes the use of variational autoencoders (VAEs), which are deep generative models, as an al-

ternative backend for spoofing detection. Different VAE modelling approaches are further studied. Furthermore, this section also studies VAEs in a classical setting — using VAEs as a feature extractor and training a separate classifier for classification. Finally, this chapter concludes with a summary in Section 5.6.

5.2 Ensemble models for spoofing detection

5.2.1 Introduction

Designing a single model to robustly detect unseen spoofing attacks can be challenging, as demonstrated at the ASVspoof 2015 and 2017 challenges, where the best performing systems [Patel and Patil, 2015, Lavrentyeva et al., 2017, Nagarsheth et al., 2017] made use of an ensemble model that combines input features or classifier scores. To this end, this section proposes ensemble models for robust spoofing detection combining both deep neural networks and traditional machine learning models through logistic regression. Our proposed approach is evaluated on the latest spoofing dataset ASVspoof 2019 (see Section 3.3) which was released as a result of the 2019 spoofing evaluations (Section 2.3). Both logical access (LA) and physical access (PA) spoofing detection tasks of the challenge are considered. Although the main focus of this thesis is on PA attacks (replay spoofing), this section describes results on the LA subtask as well because of our participation in both the subtasks of the ASVspoof 2019 challenge. The work reported here was published in [Chettri et al., 2019].

The next Subsection 5.2.2 describes our proposed ensemble model comprising various deep and shallow models. We build our models by discarding data points ensuring non-overlap in spoofing conditions between training and validation for better generalisation. For this, we propose a dataset partition which is described in Subsection 5.2.3. This dataset partition ensures that different attack types are present during training and validation to improve system robustness. Then Subsection 5.2.4 evaluates the performance of our ensemble and all the models in the ensemble. Two evaluation metrics, EER and t-DCF, as described in Section 3.5, are considered for this. We demonstrate that combining information from deep and traditional machine learning approaches along with our dataset partition can improve model generalisation. Furthermore, results on the PA tasks suggest that the same CNN performs much better when trained on the last 4 seconds of audio than on the first 4 seconds. We find that spoofed audio recordings for the PA task tend to have more silence at the end than bonafide recordings. Following this, Subsection 5.2.5 performs three different interventions proving that models exploit this silence pattern in the dataset and achieve lower performance without these cues. Our results suggest that perfor-

mance metrics reported on the current PA dataset may be overestimating the actual performance of the models, which might become somewhat of a "horse" [Sturm, 2014] that trivially sidestep the actual problem, thus raising concerns about model validity as well as performance results. Finally, Subsection 5.2.6 provides a summary of the work done in this section.

5.2.2 Models in the proposed ensemble

This section describes the approach used to design countermeasures for the LA and PA tasks of the ASVspoof 2019 challenge. A model ensemble is used in order to combine information from different countermeasure models employing various features and training procedures. This diversity leads to a powerful ensemble with good generalisation. Our ensemble comprises various deep and shallow models which are described next.

Deep Models

We train five deep models using raw audio or time-frequency representations as input to minimise a binary cross-entropy loss using Adam optimiser and early stopping with a patience of P epochs. As the dataset has more spoofed examples, we replicate the bonafide examples to ensure that each batch contains an equal number of bonafide and spoofed examples, which helps stabilise training. At inference time, we use the output layer sigmoid activation as a score. We provide model-specific training details below.

Convolutional Neural Network (CNN). We use the CNN architecture from Subsection 4.5.2 featuring 50% dropout in the fully connected layers, a batch size of 32, and a learning rate of 10^{-4}. We train the model for 100 epochs with an early stopping patience of $P = 5$ and $P = 2$ for the LA and PA tasks, respectively. We use an utterance-level mean-variance normalised log spectrogram[1], computed using a 1024-point FFT with a hop size of 160 samples, as the input. For each task, we train two such CNN models, model A and B, on the first and last 4 seconds of each audio sample. We truncate or loop the spectrogram time frames to obtain a unified time representation.

Convolutional Recurrent Neural Network (CRNN). We use a modified version of the CRNN architecture from [Morfi and Stowell, 2018] (model C). We train the model for 500 epochs with early stopping patience of $P = 10$ for both the LA and PA tasks. As input, we use a mean-variance (computed on the training

[1]Power-spectrogram for the LA task and Mel-spectrogram with 80 mel bands (for computational reasons) for the PA task.

set) normalised log-Mel spectrogram of 40 Mel bands, computed on the first 5 seconds of truncated or looped audio samples, using a 1024-point FFT with a hop size of 256 samples. During training, we use a batch size of 8 and 32 for the LA and PA tasks, respectively, with an initial learning rate of 10^{-5} that is halved on the validation loss plateau with a patience of $P = 5$ epochs, until 10^{-8}.

1D-Convolutional Neural Network. We use the network architecture from the sample-level 1D CNN [Lee et al., 2017] (model D). In total, the model consists of 9 *ReSE-2* blocks [Kim et al., 2018]. These blocks are a combination of *ResNets* [He et al., 2016] and *SENets* [Hu et al., 2018]. We use multi-level feature aggregation, where the outputs of the last three blocks are concatenated and followed by a fully connected layer of 1024 units, batch normalization and ReLU layers, a 50% dropout layer and a fully connected layer of 1 unit with sigmoid activation. Each convolutional layer has filters of size 3, an *L2* weight regularizer of 0.0005, and all strides are of unit value. The raw audio input is 3.7 seconds in duration and randomly sampled segments of this size are selected from the recordings. We loop shorter samples to obtain a unified time representation. We train the model using a batch size of 16, learning rate of 10^{-4} and an early stopping patience of $P = 25$ epochs.

Wave-U-Net. We use a modified version of the Wave-U-Net [Stoller et al., 2018], with five layers of stride four, and without upsampling blocks (model E). The outputs of the last convolution are max-pooled across time, reducing the parameter count and incorporating the intuition that the important features in the tasks are temporally local. Finally, we apply a fully connected layer with a single output to yield a classification probability. We train the model using a batch size of 64, a learning rate of 10^{-5} and early stopping patience of $P = 10$ for both the LA and PA tasks, where an epoch is defined as 500 update steps. To ensure the audio inputs have the same length, we pad all recordings with silence to 196608 audio samples (= 12.23 seconds). For the PA task, we also match real samples to their spoofed versions based on the speaker identity and utterance. We train on pairs of audio samples (discarding samples without any matches) and balanced batches, in order to stabilise the training process and improve generalisation by preventing the network from using speaker identity and utterance content for discrimination.

Shallow Models

Additional to deep models, we use two different shallow [Salamon et al., 2017] models: Gaussian Mixture Models (GMMs) and Support Vector Machines (SVMs).

GMM. We train three GMM models using MFCC (model F), IMFCC (model G), and SCMC (model H) features due to their performance on the ASVspoof 2015 [Sahidullah et al., 2015] and 2017 spoofing datasets (Section 4.2.2). For each of them, we extract 60 dimensional static-delta-acceleration (SDA) feature vectors per frame. Section 2.5 provides background information on these features. We use 128 and 256 mixture components for the LA and PA tasks respectively and train one GMM each for the bonafide and spoof classes. At test time, the score of each test utterance is computed as the log likelihood ratio between the bonafide and spoofed GMM model as described in Equation 4.1. We use the publicly available scripts and feature configuration from [Sahidullah et al., 2015] for computing these features.

SVM. We train two SVMs using i-vectors (model I) and the long-term-average-spectrum (LTAS) feature (model J) since they have shown good performance on prior spoofing datasets [Sahidullah et al., 2015, Muckenhirn et al., 2017a]. Inspired by [Novoselov et al., 2016b], we fuse multiple i-vectors in our approach, each based on complementary hand-engineered features, and manage to improve performance over a single i-vector based SVM. We train four different i-vector extractors using MFCC, IMFCC, CQCC and SCMC features. For each of them we use a 60-dimensional SDA feature vector per frame. We train the T matrix with 100 total factors on both tasks and a universal background model (UBM) with 128 and 256 mixtures on the LA and PA tasks, respectively and extract 4 different 100-dimensional i-vectors for every utterance. We use 400-dimensional fused i-vectors for the LA task and 300-dimensional fused i-vectors for the PA task. We perform mean-variance normalisation on the fused i-vectors and LTAS feature and train SVMs with a linear kernel and the default parameters of the Scikit-Learn [Pedregosa et al., 2011] library. We train the UBM and T matrix using the MSR-Identity toolkit [Sadjadi et al., 2013].

Ensemble

We investigate several ensemble models combining different sets of deep and shallow models. We optimise this choice monitoring the performance on a subset of the development data (***dev_es***, details in the next section). We do not use this subset during the training of fusion weights. The weights are learned using a logistic regression implementation using the Bosaris [Brümmer and de Villiers, 2013] toolkit. Therefore, we only report the best ensemble setting we obtain from many combinations investigated. To this end, we define three ensemble

models E_1[2], E_2, and E_3 for both the LA and PA tasks.

The E_1 ensemble focuses on adding as many individual models (deep and shallow) as possible to include more diversity that may help improve performance on the unseen test set. On the LA task, it combines models A, C through G and I. On the PA tasks, E_1 fuses all single models except D. For the LA task, models B, H and J were not used in E_1, while model D was discarded for the PA task since including it deteriorated the ensemble performance on ***dev_es***.

E_2 on the other hand combines only deep models aiming to understand how models trained with different architectures, input representations and training strategies affect performance and generalisation on the test set. On the LA tasks, E_2 combines models A, B and E. On the PA tasks, E_2 combines all five deep models A through E. Including models C and D deteriorated the performance on ***dev_es***, so they were not included in LA.

The final ensemble E_3 combines two deep models A and B. While both models use the same architecture, they operate on different parts of the audio input —the first and the last 4 seconds audio for A and B respectively. This ensemble aims to understand how overall performance improves when the whole input is re-combined through score fusion.

5.2.3 Dataset and proposed partitions

We used the ASVspoof 2019 LA and PA datasets for evaluation of our ensemble models. Section 3.3 provides a description of these datasets. Here, we describe our proposed dataset partitions that were used during our ASVspoof 2019 challenge participation. The training and development subsets have similar spoofing algorithms/conditions in both the LA and PA datasets. We argue that using the same types of spoofing attacks during training and validation might lead to overfitting and poor generalisation on unseen attack conditions. Thus, we further partition the original training and development datasets for both LA and PA, ensuring non-overlap in spoofing attack conditions. Protocol files and details related to data partitions are available in `https://github.com/BhusanChettri/ASVspoof2019`.

LA dataset partition. The spoofed utterances in the training and development sets are generated using one of the four speech synthesis algorithms: SS_1, SS_2, SS_4, US_1 and two voice conversion algorithms: VC_1 and VC_4. The voice conversion algorithms are based on neural network (VC_1) and transfer function based methods (VC_4) [Bonastre et al., 2006]. Speech synthesis

[2]The ensemble E_1 was the primary system we submitted to the ASVspoof 2019 evaluations for ranking.

algorithms are based on waveform concatenation (US_1) [Morise et al., 2016], neural network-based parametric speech synthesis using Wavenet (SS_1) [Oord et al., 2016], neural network-based parametric speech synthesis using source-filter vocoders (SS_2) [Morise et al., 2016], and publicly available toolkits such as Merlin[3], CURRENT[4] and MaryTTS[5] (SS_4).

We create the ***train_tr***[6] subset from the original training set by discarding all the spoofed utterances for the SS_1 and VC_1 spoofing conditions. However, all the bonafide utterances of the training set are used in ***train_tr***. Therefore, our proposed training partition ***train_tr*** consists of 2580 bonafide and 15200 spoofed utterances.

We partition the original development set into two subsets: ***dev_es*** and ***dev_lr***. We use ***dev_es*** for model validation, early stopping and parameter tuning; and the ***dev_lr*** partition is used to learn fusion weights through logistic regression. In the ***dev_es*** partition we use the two spoofed conditions SS_1 and VC_1 that were not used in ***train_tr***. Among 20 speakers in the bonafide class of the development set, we chose 12 speakers (8 female and 4 male speakers) randomly for the bonafide class of this partition. Furthermore, only 10 speakers out of these 20 speakers in the development set are seen in the spoofed class. Therefore, we use 6 speakers (4 female and 2 male) out of 10 in the ***dev_es*** partition. Following this criterion, the ***dev_es*** partition comprises 5160 spoofed files corresponding to the two spoofing attack conditions and six speakers and 1820 bonafide files.

In the ***dev_lr*** partition we keep all the six attack conditions of the development set but ensure that speakers seen in the ***dev_es*** partition are not used here to avoid overfitting on speakers. Therefore, in the bonafide class of this partition, we use those 8 speakers (4 male and 4 female) that we discarded in the ***dev_es*** partition. This gives us 728 bonafide files. As for the spoofed class, we consider all 6 attack conditions but use only those 4 speakers that were discarded in the spoofed class of the ***dev_es*** partition. Therefore, the ***dev_lr*** partition comprises 728 bonafide and 6816 spoofed utterances.

PA dataset partition. Unlike LA, the attack conditions/configurations in PA are defined in a different manner. Here, we combine the environment identifier and attack identifier as replay attack conditions on which we define our dataset partition. In other words, we represent the attack condition as a quintuplet: S,R,D_s,D_a,Q where the first three represent the environment ID and the re-

[3] `https://github.com/CSTR-Edinburgh/merlin`

[4] `https://github.com/nii-yamagishilab/project-CURRENNT-public`

[5] `http://mary.dfki.de`

[6] The suffixes ***tr***, ***es***, and ***lr*** signifies our proposed partition used for model training, early stopping (model validation) and learning ensemble model through logistic regression.

maining two the attack ID. Parameters S,R,D_s represents room size in square meters, reverberation and talker-to-ASV distance in centimeters. Parameters D_a and Q represent attacker-to-talker distance in centimeters and replay device quality (low, medium and high quality). Following this, we get a total of 243 different attack settings. Both the training and development sets have these attack settings. As in LA, we follow the same procedure and ensure that no attack conditions or speakers are overlapping across training and model validation and while learning fusion weights through logistic regression.

Therefore, in ***train_tr*** we keep 170 attack conditions (chosen randomly) but all the bonafide files from the training set. This gives 5400 bonafide files and 33700 spoofed files in this partition. For early stopping and model validation the ***dev_es*** partition uses 73 attack conditions different from the ones used in ***train_tr***. This gives 4050 bonafide files and 5966 spoofed files in ***dev_es*** partition. As for the ***dev_lr*** partition we use all the 243 attack conditions but non overlapping speakers from ***dev_es***. This gives 1350 bonafide and 4860 spoofed files in the ***dev_lr*** partition.

Please see [Todisco et al., 2019.] and the ASVspoof 2019 evaluation plan[7] for further details on the acronyms and different attack conditions for the LA and PA datasets discussed here.

5.2.4 Evaluation

This section describes the approach used in model training following the proposed dataset partition in Subsection 5.2.3. A brief description of the metrics considered for performance evaluation is provided. It then evaluates the effectiveness of our proposed data protocols for training and model validation. Then it describes the performance of all our single and ensemble models on both the development and evaluation sets for both the LA and PA tasks.

Training and testing

We train our models (single and ensemble) described in Subsection 5.2.2 using the ***train_tr*** and ***dev_lr*** sets respectively. We use ***dev_es*** for model validation, early stopping and hyper-parameter optimisation. We compare our models' performance with the baseline LFCC (model B_1) and CQCC (model B_2) feature based GMM models provided by the ASVspoof 2019 challenge organisers.

Performance metric

We use the EER and t-DCF metrics to evaluate model performance. Section 3.5 and Section 2.3 provide details on these metrics and the ASVspoof 2019 chal-

[7] `https://www.asvspoof.org/asvspoof2019/asvspoof2019_evaluation_plan.pdf`

Table 5.1: Comparison of Model A (CNN) performance trained and validated using the original protocol (*) and our proposed protocol.

Protocol	Test set*	LA		PA	
		t-DCF	EER%	t-DCF	EER%
Proposed	Dev	0.0074	0.32	0.2795	**10.77**
	Eval	**0.1790**	**7.66**	**0.3091**	**12.16**
Original	Dev	**0.0**	**0.0**	**0.2693**	10.94
	Eval	0.3599	11.26	0.3161	12.36

lenge, respectively.

Effectiveness of the proposed data partition

To evaluate the effectiveness of our proposed protocol for training and model validation, we train a deep countermeasure model using both our proposed and the original protocols. For this, we take Model A which operates on spectrogram inputs. We chose this model architecture as it takes the least computational time (training and testing) among other deep architectures we considered in this section. Furthermore, our objective here is to confirm and validate our hypothesis towards using distinct spoofing attack conditions during training and model validation for better generalisation.

Table 5.1 summarises the performance of the CNN on the original development and evaluation sets. The results clearly indicate that avoiding the use of same attack conditions during model training and validation indeed helps improve model generalisation. On the LA tasks, this is more prevalent as we can observe that the CNN trained on the original protocol overfits easily on the development set showing poor generalisation on the evaluation set. As for the PA tasks, we witness improved generalisation using our protocols, but the gain is not as substantial as we saw in the case of LA. This also suggests the difficulty of the replay spoofing detection task itself. Nonetheless, it should be noted that our proposed protocol discards a lot of data points yet achieves improved performance on the evaluation set in comparison to using the original data protocols.

Development set results

Table 5.2 summarises the results on the original development set for both the LA and PA tasks. In general, the results suggest that the PA task is harder than the LA task. For the PA task, our CNN performs noticeably better when operating on the last 4 seconds of audio (model B) instead of the first 4 seconds (model A), suggesting the presence of discriminative cues at the end of each audio signal which we confirm in Subsection 5.2.5. Furthermore, we observe

Table 5.2: Results on the LA and PA development sets. Bold: best performance, na: not applicable. A and B uses the same CNN but are trained on the first and the last 4 seconds spectrogram respectively. * as we do not use the same set of models for LA and PA, we do not provide the detailed model combination here. Please see Subsection 5.2.2 for ensemble model details.

Model	Model ID	LA		PA	
		t-DCF	EER%	t-DCF	EER%
LFCC-GMM	B1	0.0663	2.71	0.2554	11.96
CQCC-GMM	B2	0.0123	0.43	0.1953	9.87
CNN	A	0.0074	0.32	0.2795	10.77
CNN	B	0.0040	0.27	0.1672	5.98
CRNN	C	0.1706	5.65	0.1223	5.0
1D-CNN	D	0.36	13.58	0.9269	36.28
Wave-U-Net	E	0.0745	2.43	0.4725	21.16
MFCC-GMM	F	0.1805	7.46	0.2354	10.88
IMFCC-GMM	G	0.0438	1.73	0.2119	8.94
SCMC-GMM	H	*na*	*na*	0.2787	12.46
i-vector SVM	I	0.0045	0.16	0.2537	9.93
LTAS SVM	J	*na*	*na*	0.3534	13.6
Deep and shallow*	E1	**0.0**	**0.0**	**0.0354**	**1.33**
Only deep	E2	0.0002	0.03	0.0523	1.85
A+B	E3	0.0025	0.2	0.1316	4.85

a poor performance for models D and E. Apart from having to learn features directly from the raw audio, another reason could be that they involve zero-padding all signals or using a randomly selected audio segment for prediction, respectively, and thus might not be able to exploit such cues at the end of audio signals.

Our i-vector feature fusion approach (model I) shows impressive performance on the LA task but relatively poor performance on the PA task. One reason for this could be that the i-vectors extracted using hand-crafted features are not able to capture characteristics of unseen replay attack conditions. On both the LA and PA tasks, model G (IMFCC) outperforms model F (MFCC) showing consistency on prior findings on the v1.0 and v2.0 of the ASVspoof 2017 datasets (Subsection 4.2.2 and Subsection 4.4.3). This suggests that a focus on higher frequency information is beneficial as it might not be perfectly generated by the TTS and VC algorithms. Likewise, on the PA task, the playback device properties may impact high-frequency content. Finally, the poor performance of models H and J suggest that the SCMC and LTAS features are not suitable for this task.

As expected, our ensemble model appears to benefit from combining different models for both tasks, as indicated by the strong reduction in t-DCF and EER compared to all individual models. On both tasks, E_1 performs better than E_2

Table 5.3: Results on the LA and PA evaluation sets.

Model	LA		PA	
	t-DCF	EER%	t-DCF	EER%
B1	0.2116	8.09	0.3017	13.54
B2	0.2366	9.57	0.2454	11.04
A	0.1790	7.66	0.3091	12.16
B	0.3841	19.11	0.1577	5.75
E1	**0.0755**	**2.64**	0.1492	6.11
E2	0.2136	9.57	0.2913	14.12
E3	0.2952	10.63	**0.1465**	**5.43**

which in turn performs better than E_3.

Evaluation set results

Table 5.3 summarises the results on the evaluation set[8]. On the LA task, model E_1 has an EER of 2.64% and a t-DCF of 0.0755, outperforming the baselines by a large margin and securing the third rank in the ASVspoof 2019 challenge. The superior performance of E_1 over E_2 and E_3 suggests that fusing multiple models employing different features does provide complementary information useful for spoofing detection.

However, on the PA tasks our single model B outperforms ensemble models E_1 (on the EER) and E_2 (both metrics). Furthermore, our two model ensemble E_3 (A+B) outperforms the five deep model ensemble E_2 and nine model ensemble E_1 reaching the lowest t-DCF of 0.1465 and an EER of 5.43%. While these results suggest good model generalisation, they raise questions about the relevance of the cues used by model B as it is only trained on the last 4 seconds of each recording. Besides the poor performance of models D and E, the inferior performance of ensemble models on the evaluation set compared to the development set (Table 5.2) could be explained by model C making random predictions on the evaluation data (due to a bug we found after the challenge submission), but not on the development set – which is corroborated by the fact that model C receives the second highest weight by logistic regression in both E_1 and E_2.

Furthermore, post-evaluation, having received the labels for the evaluation set, we rescored all our individual systems and our primary ensemble system E1 using the corrected system C scores. Table 5.4 summarises the results. On the LA subtask our updated system E1 do not seem to have a substantial impact, however, on the PA subtasks we observe a significant impact.

[8]Computed by the ASVspoof 2019 challenge organisers.

Table 5.4: Results on the LA and PA evaluation set scored post-evaluation after the evaluation set labels were released by the ASVspoof organiser. * the corrupted system that was used during the challenge submission. Updated: rescored after using the correct scores from system C.

Model	LA		PA	
	t-DCF	EER%	t-DCF	EER%
C*	0.9993	49.62	0.9999	50.31
C (updated)	0.2536	10.45	0.2003	7.25
D	0.3801	14.88	0.6087	24.79
E	0.1841	7.81	0.9838	42.91
F	0.2988	11.41	0.3151	12.82
G	0.2615	12.35	0.2560	10.66
H	*na*	*na*	0.3311	14.05
I	0.2912	11.16	0.2413	9.415
J	*na*	*na*	0.3768	15.25
E1 (submitted)	**0.0755**	**2.64**	0.1492	6.11
E1 (updated)	0.0878	3.24	0.0620	2.36

5.2.5 Interventions on the PA tasks

In Table 5.2 we find that for the PA task, the same CNN performs much better when trained on the last 4 seconds of audio (model B) than on the first 4 seconds (model A). We thus analyse a set of audio recordings for the PA task that were confidently classified by model B and find that spoofed audio tends to have more silence (zero-valued samples) at the end than bonafide examples. In comparison, silence at the beginning of the recordings is often shorter and does not appear to follow this pattern. Therefore, we hypothesize that any model (deep or shallow) trained on the PA dataset that does not specifically discard this information could exploit the duration of silence as a discriminative cue. This leads to countermeasure models that are easily manipulated, simply by removing silence from the spoofed signals to make the model misclassify them as a bonafide signal, and vice versa. To demonstrate this effect in practice, we perform three interventions on model B and the adapted[9] baselines M_1 and M_2 by manipulating the silence at the end of the audio signal.

Intervention I: During testing

In this intervention we train the models on the original recordings with silence but remove silence during testing[10]. In Table 5.5, a strong increase can be noticed in both EER and t-DCF for all models, suggesting that they indeed rely on the silence parts for prediction. We find that model B is most sensitive

[9]We use 128 mixtures to train the LFCC (M_1) and CQCC (M_2) GMMs in contrast to 512 mixtures used in the baselines B_1 and B_2.

[10]We use a naive approach of counting the first consecutive zeros as silence and remove them.

Table 5.5: Intervention results on the *development set* of the PA tasks. Numbers to the left of the arrow indicate performance without any intervention.

Intervention	Model	t-DCF	EER%
I	M1	0.2036 → 0.2741	9.18 → 13.27
	M2	0.1971 → 0.2959	10.06 → 15.59
	B	0.1672 → 0.5018	5.98 → 19.8
II	M1	0.2036 → 0.9528	9.18 → 54.76
	M2	0.1971 → 0.9463	10.06 → 57.98
	B	0.1672 → 0.2626	5.98 → 11.20
III	M1	0.2036 → 0.8614	9.18 → 41.09
	M2	0.1971 → 0.9448	10.06 → 58.71
	B	0.1672 → 0.3129	5.98 → 12.85

Table 5.6: Same as in Table 5.5 but for the *evaluation set*.

Intervention	Model	t-DCF	EER%
I	M1	0.2483 → 0.3349	10.85 → 15.07
	M2	0.2498 → 0.3698	11.07 → 16.55
	B	0.1576 → 0.5263	5.75 → 19.75
II	M1	0.2483 → 0.9734	10.85 → 53.34
	M2	0.2498 → 0.9923	11.07 → 56.84
	B	0.1576 → 0.2926	5.75 → 11.7
III	M1	0.2483 → 0.8871	10.85 → 38.97
	M2	0.2498 → 0.9928	11.07 → 59.12
	B	0.1576 → 0.3472	5.75 → 13.98

to this intervention, with t-DCF and EER rising by 0.3346 and an absolute 13.82%, respectively. This could be due to deep models focusing more strongly on silences than the GMM models, which are trained on individual spectral frames and aggregate the score through averaging frame-wise likelihoods.

Intervention II: During training

Here, we train the model with silence parts removed, but test on the original test recordings (with silence). The stable performance of the CNN (model B) over the GMMs in Table 5.5 suggests that the former is more robust against variations in silence duration. On the other hand, we find a dramatic increase in the error rates for M_1 and M_2. One interpretation for this is that the bonafide and spoof GMMs may assign a low likelihood to silence frames as they have not seen them during training. Thus, silence frames do not make large contributions to the final score making the task much harder.

Intervention III: During training and testing

In this intervention, we remove silence during training and testing to ensure that the audio samples do not share an easily exploitable cue. This forces the

models to learn about the actually relevant factors of interest and thus provides more realistic performance estimates (Table 5.5). As in intervention II, model B shows a stable performance indicating good generalisation and discrimination capabilities. Models M_1 and M_2 on the other hand achieve poor performance, possibly since their bonafide GMM models assign a high likelihood to spoofed frames as they are very similar to bonafide ones when only considering the speech frames.

Finally, we repeat the above three intervention experiments on the evaluation set. Table 5.6 summarises the results. Since spoofed files in the evaluation set have similar issues of silence as in the development set, we observe similar trends across the three different intervention experiments.

5.2.6 Discussion

This section proposed an ensemble modeling approach towards the logical access (TTS and VC) and physical access (replay) spoofing detection problem on the ASVspoof 2019 dataset (Section 5.2.2). Then Subsection 5.2.3 described our proposed dataset partitions for training (***train_tr***), validation (***dev_es***) and learning fusion weights (***dev_lr***). This partition involves discarding a lot of spoofed data points to ensure that there is no overlap in terms of spoofing attack conditions and speakers between these sets. In Subsection 5.2.4 we evaluated and demonstrated (see Table 5.1) the effectiveness of our proposed data partition by comparing the generalisation performance of our CNN model when it is trained using the original and our proposed protocols. Following this, the experimental results in Subsection 5.2.4 further showed that combining models trained on different feature representations and using our proposed dataset partition can be effective in detecting unseen spoofing attacks. We achieve good performance on the PA task and 3^{rd} ranking on the LA task of the ASVspoof 2019 challenge. The PA task seems generally more difficult and will be the primary focus of future work. Our intervention experiments described in Subsection 5.2.5 suggest that many models trained on the PA dataset can become somewhat of a "horse" [Sturm, 2014], where solving the actual problem is unintentionally avoided by exploiting silence as trivial cues. As the evaluation set also contains such silences (see Table 5.6), the reported performance metrics in this task currently overestimate the actual performance. In addition to removing silence from the end of recordings, we also removed it from the beginning, but found lesser impact on the performance and therefore do not report those results in this section. Overall, the work in this section demonstrated the effectiveness of choosing a good data partition to use during model training and validation. Although this might mean losing a lot of data points, our experi-

mental results demonstrated that such approach can potentially improve model robustness.

5.3 Overcoming the impact of dataset artefacts

5.3.1 Introduction

Artefacts and confounders in a dataset can bias decisions of a machine learning model making them unreliable and untrustworthy. Such artefacts, if appearing in a dataset, could be the result of methods used in data collection, compilation, aggregation and partition [Rosset et al., 2010]. These issues can occur in any data-driven machine learning task, and the impact caused by such biases in domains such as medicine, finance and security is not affordable. Therefore, it is important to provide some mechanism (eg., a pre-processing step) that would help mitigate the impact of biases induced by such dataset artefacts in machine learning decisions. A background on artefacts and their impact on ML is provided in Subsection 2.9.1. As suggested in Section 2.9, the results of a trustworthy ML model should be independent of the factors or cues in a dataset that are not relevant to the problem. As summarised in Subsection 2.9.1, the learning algorithms can easily exploit artefacts and confounders (if present) within the training data. Biases introduced by such confounding factors often contribute towards solving the problem measured using some figure of merit (EER, for example). Apparently, as explained in [Hernandez-Orallo, 2019, Sturm, 2014] these ML models behave much like a "horse" in machine learning since they provide excellent results using cues not relevant to the actual problem [Sturm, 2016, Rodríguez-Algarra et al., 2019].

To this end, this thesis has identified "horses" in machine learning for anti-spoofing research applied to both the ASVspoof 2017 and ASVspoof 2019 PA benchmark datasets. Section 4.2 described how initial silence in some of the bonafide recordings served as a potential cue on the version 1.0 of the ASVspoof 2017 dataset. As a result, an updated version 2.0 dataset was released fixing these issues. However, as explained in Section 4.6 the potential cues/confounders in this dataset were not completely removed. The experimental intervention experiments confirmed the presence of artefacts and how they contributed in model predictions. Furthermore, as described in Subsection 5.2.5, on the ASVspoof 2019 PA dataset, the duration of silence at the end of audio signal served as a potential cue that models exploited for spoofing detection. A simple preprocessing step that removed silence samples from the start and end was proposed that helped mitigate the issue.

Motivated from these, the work described here builds upon Section 4.6 and aims at proposing methods to build trustworthy CM models on the ASVspoof 2017 v2.0 dataset towards producing reliable performance estimates. The next Subsection 5.3.2 explains our proposed method that uses speech endpoint detec-

tion to remove audio samples before and after the actual speech utterance. This is applied during both training and testing. As the reliability of our proposed method highly depends on the correctness of speech endpoints, we have manually prepared speech endpoint annotations for all the audio files in the training and development sets of this dataset. Therefore, in this section we train our CM models applying these annotations during training and validation, which ensures that parameter updates and model selection are independent of artefacts that were found in the start and end of audio recordings as highlighted in Subsection 3.2.3. During testing, we use endpoint detection derived using automatic speech activity detection. We use automatic methods on the evaluation set for two reasons. First, due to time constraints, manually annotating the large amount of audio recordings (more than $13,000$ recordings) was not possible. Second, even if we do not get accurate endpoint annotations on the evaluation set, this would not have a substantial impact on performance as models have been trained and validated using the manual endpoint annotations. The section also provides a description of both manual and automatic endpoint annotations. Then the next Subsection 5.3.3 evaluates our proposed methodology using the same five different countermeasure models that were studied in Section 4.6. In addition, a novel frame-level deep CM model is proposed based on endpoint detection that demonstrates robust performance even in the presence of recording artefacts at test time. New benchmark results are provided for all of these CM models that could serve as new baselines. The effectiveness of our proposed approach is further demonstrated by running the same intervention experiment from Subsection 4.6.4 using the same "burst click sound" (BCS) artefact. Finally, Subsection 5.3.4 provides a summary of the work done in this section.

5.3.2 Proposed method

We now describe our proposed methodology to address the issues highlighted in the ASVspoof 2017 v2.0 dataset (Subsection 3.2.3). To this end, we propose the use of speech endpoint detection during training and inference to build reliable and trustworthy CMs on this dataset. Fig. 5.1 illustrates our proposed idea. The first three blocks are shared during training and testing. The endpoint detection module removes raw samples before and after the actual speech utterance. This ensures that both bonafide and spoof utterances now have similar audio patterns and are free from recording artefacts we highlighted in Section 3.2.3. The cleaned speech signal is then passed on to the subsequent modules for feature extraction, model training and inference. As before (Subsection 4.6.3) the role of *unify duration*, an optional module remains the same. It truncates

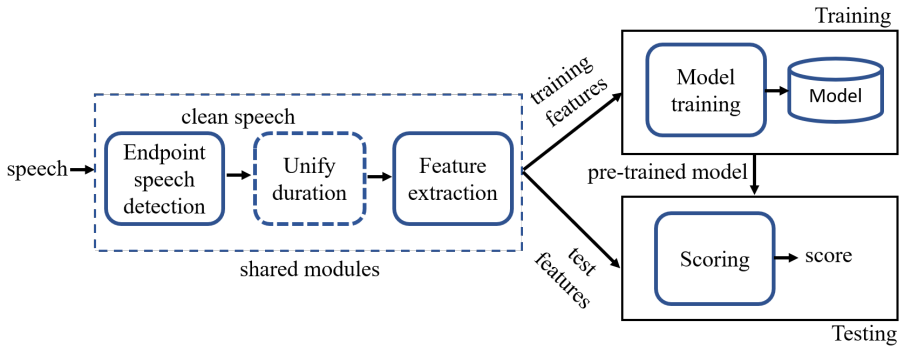


Figure 5.1: Proposed CM design for trustworthy performance estimates.

or replicates audio samples to create a fixed-duration input representation, and is applicable only for the CNNs. Finally, we also propose a frame-level deep countermeasure model (DNN) with endpoint detection for robust performance.

In the following we briefly discuss the approach used for speech endpoint detection. Then we evaluate the performance of our new models providing new benchmark results. Finally we demonstrate the effectiveness of our proposed method through BCS intervention experiments and compare its robustness with the initial models (Subsection 4.6.2). Furthermore, from hereon we discard the use of corrupted audio files identified in Subsection 3.2.3 during training and testing.

Speech endpoint detection

We use two approaches for speech endpoint detection: manual and automatic. The manual approach uses speech endpoint annotations that we collected during the dataset inspection. Automatic speech endpoint detection is based on rVAD [Tan et al., 2020], a robust voice activity detection algorithm.

Manual endpoint annotation. We manually inspected all the training and development set audio files and bonafide audio files in the evaluation set of the ASVspoof 2017 v2.0 dataset. We record the speech start and end points for them during the inspection. All audio recordings were carefully listened to mark the annotations, and the process was often repeated for the replayed recordings to ensure correctness. We further validated these annotations and updated a few erroneous annotations. We stress that due to time constraints and a large number of spoof files in the evaluation set we did not carry out a manual inspection on them. These annotations were developed by the author himself using Audac-

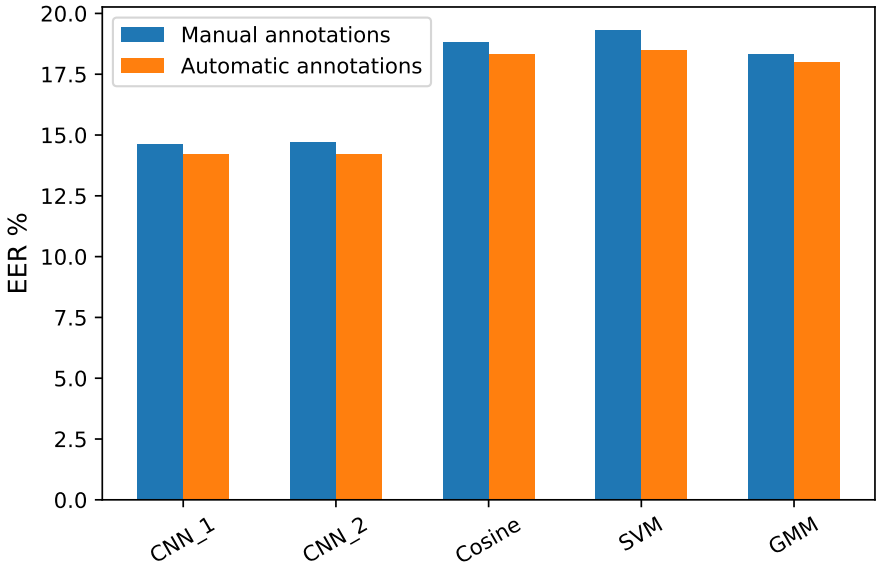


Figure 5.2: Performance (EER%) on the evaluation set (ASVspoof 2017 v2.0) using models trained with manual and automatic speech endpoint annotations.

ity[11]. One of the key motivations for manual annotations is to ensure reliable endpoints which may be challenging using automatic methods due to the nature of the audio recordings in this dataset. We wanted to ensure that the annotations for the training and development sets are as accurate as possible because model parameters are trained and optimised on them.

Automatic speech endpoint detection. We also create speech endpoint annotations using an automatic voice activity detection (VAD) algorithm. In particular, we use rVAD [Tan et al., 2020] – robust voice activity detector – that operates on raw audio samples. The main reason for choosing the rVAD algorithm is that this approach has shown promising results in noisy conditions. Moreover, this algorithm focusses on robust voice activity detection in the presence of burst-like noise sound that has high energy. In the first step, high-energy segments are detected using the a posteriori SNR weighted energy difference measure. Then, the method checks for pitched segments. If found, they are regarded as speech else all samples are set to zero and the segment is labelled as nonspeech [Tan et al., 2020]. As the ASVspoof 2017 dataset contains noisy speech recordings and recordings with burst-click sounds, the use of this algorithm suits our problem. Below we describe the steps we used to derive speech endpoints using this algorithm.

[11]https://www.audacityteam.org/

1. Framing: using a 25 ms frame window and a frame shift of 10 ms, the original audio signal is split into frames. Then, we run the rVAD algorithm on these frames to obtain a sequence of binary labels (0's and 1's).

2. Mark the speech start and end points using the following steps:

    - We use **N** consecutive frames of 1's as speech onset detection threshold, and mark its first frame as the speech start point.
    - Similarly, the algorithm counts for the detection of **N** consecutive frames of 0's after the speech onset. If found, its first frame is marked as endpoint.
    - Derive the start and end times from the speech onset and offset frame markers.

We tried different values for **N**, but we chose to use $\mathbf{N} = 10$ that accounts for 100 ms samples as the onset speech detection threshold because this showed better approximation when compared with ground truth annotations (using manual endpoint annotations) on the training and development set. Finally, we validate the accuracy of the automatic endpoint speech annotations by comparing them with our manual annotations on the development set. Using a 200 ms collar, the speech onset and endpoint accuracy is 97% and 85% respectively. Further, with a 100 ms collar these accuracies were 68% and 56% respectively.

Finally, we also compare the performance estimate on the evaluation set using CM models using both the manual and automatic annotations during training and validation. Fig. 5.2 summarises this result. We find a comparable performance using models trained using manual and automatic annotations. This holds true across all five CM models, confirming the reliability of r-VAD method for computing speech endpoint annotations in this dataset. Therefore, we use manual speech endpoint annotations for training and validating models ensuring the correctness of our approach. Following the reliable performance of our models trained and tested using automatic speech endpoint annotations (see Fig. 5.2) we use automatic speech endpoint detection during testing.

Frame-level deep countermeasure model

From our study so far, it is confirmed that deep CMs (CNNs) not only show superior performance over other CMs, but are equally more sensitive to artefacts in this dataset. One reason for this accounts to the fixed-duration input representation used by them. Duplicating audio contents to match the desired duration also involves spreading artefacts in the audio signal (see Fig. 4.6). As a result they become more sensitive to artefacts (see Tables 4.26, 4.27), and hence they are less reliable and untrustworthy.

Motivated from this, we propose a frame-level deep CM model (DNN) that is trained on the original audio contents without requiring to truncate or copy audio samples as in CNNs. The use of context-frames — augmenting past and future time frames to the current time frame — is often adopted in training frame-level DNNs [Cai et al., 2017]. However, for direct comparison with GMMs we do not use any context-frames in this work. This DNN treats the inputs as a bag-of-frames much like the way GMMs are trained. Its architecture comprises a series of fully connected layers and operates on a single input frame to predict whether the frame corresponds to a bonafide or spoofed class. The details of the DNN architecture is provided in Appendix A.

As for training, we use the same procedure that was used in training the CNNs (Subsection 4.5.2) but with different input representation. The input to the DNN is a spectrogram frame of shape 1×257, where 1 corresponds to time frame and 257 to the number of frequency bins computed using a 512 point FFT with a 10 ms hop. During testing, for a given test utterance, we compute the score for every frame and take their average as the final score.

For comparison and completeness of the study, we first train and evaluate this DNN on the original dataset (without endpoint detection). Our DNN reports an EER of 28.95% on the the evaluation set which is worse than CNN_1 (10.7%) and CNN_2 (13.4%, see Table 4.23) trained using fixed-input representations. Although CNNs trained with context information yield better detection performance over frame-based DNNs, we demonstrate in the following subsection that DNNs are much more robust and trustworthy than CNNs.

5.3.3 Experimental setup and evaluation

This section describes different features, classifiers and performance metrics considered to evaluate CM models trained using speech endpoint detection. New benchmark results for both frame-level and utterance-level CM models are described. Finally, this section also demonstrates the robustness of the proposed method across various CM models confirming the reliability of endpoint detection as a simple method to mitigate the issues found in the ASVspoof 2017 v2.0 dataset.

Features, classifier and performance metric

We use the same input features (CQCCs, i-vectors and power spectrograms) and backend classifiers (GMMs, Cosine Distance, SVMs and two CNNs) from Subsection 4.6.2 to evaluate our proposed method. We train these CM models in a similar way as described in Subsection 4.6.2 but now we apply the endpoint detection that discards everything before and after the actual speech utterance.

As for the performance evaluation of these CMs we use the EER metric as described in Subsection 3.5.1.

Table 5.7: New benchmark results (EER %). (1) Condition 1: evaluate on the original test dataset. (2) Condition 2: same as in (1) but uses automatic endpoint detection.

Model	Set	Condition 1	Condition 2
CNN_1	Dev	7.76	9.0
	Eval	17.2	14.58
CNN_2	Dev	8.6	9.49
	Eval	15.16	14.77
Cosine	Dev	14.76	15.91
	Eval	20.49	18.89
SVM	Dev	14.80	15.76
	Eval	21.34	19.26
GMM	Dev	16.41	16.21
	Eval	18.6	18.29
DNN	Dev	13.03	12.92
	Eval	17.55	15.94

New benchmark results

We now train and validate all our CMs applying our manual endpoint annotations. We evaluate their performance under two test conditions. Condition 1: we evaluate them on the original test data containing recording artefacts. Condition 2: we evaluate them applying automatic endpoint detection during testing. Table 5.7 summarises the results. As expected our new models now show worse performance in comparison to initial models (see Table 4.23) trained without endpoint detection. However it should be noted that our main focus here is not on improving EERs. We aim towards building trustworthy CM models providing reliable performance estimates, and making them secure from being manipulated using artefacts/cues in the dataset.

Overall deep CM models show better performance on both the development and evaluation sets compared to other CMs under both test conditions. This demonstrates their superiority in learning discriminative features. We find that all our CMs show better generalisation on the evaluation set using endpoint speech detection (condition 2). Furthermore, the small performance difference of these CMs between the two test conditions suggest that they are now less sensitive to the dataset artefacts. These results demonstrate that CMs trained on cleaned data (using endpoint detection) are more robust even in the presence of noisy test data. We provide experimental evidence in the next section to prove the robustness of our approach.

Model robustness

We now demonstrate the robustness of our newly trained CMs against BCS signatures through an intervention experiment illustrated in Fig. 4.5. We perform this intervention on all the test recordings in the development and evaluation sets using the same 100 ms BCS signature from Section 4.6.4 with one major difference. Here the intervention module performs two tasks. First it applies an automatic speech endpoint detector to remove raw samples before and after the actual speech utterance. Second it appends the BCS signature at the start of the cleaned speech signal. The updated signal is then passed to the subsequent modules for feature extraction and scoring. We score them using both our newly trained models and initial models. Finally we compare and contrast their performance in terms of EER. Table 5.8 summarises the results. Numbers shown to the left of the arrow are the results of our initial models from Table 4.23 and Table 5.7 (condition 2) is included for better readability. It should be noted that we take all our pretrained initial models from Subsection 4.6.2 to run this intervention.

Table 5.8: Model robustness experimental results. Numbers to the left and the right of arrow indicate EER% before and after the intervention on test signals using the BCS signature.

	Set	New model	Initial model
CNN_1	Dev	9.0 → 10.08	7.7 → 34.5
	Eval	14.58 → 18.01	10.7 → 36.19
CNN_2	Dev	9.49 → 7.85	7.37 → 8.25
	Eval	14.77 → 20.96	13.4 → 22.6
Cosine	Dev	15.91 → 15.24	10.6 → 15.11
	Eval	18.89 → 19.11	14.8 → 18.13
SVM	Dev	15.76 → 15.43	10.8 → 15.84
	Eval	19.26 → 19.33	15.6 → 18.84
GMM	Dev	16.21 → 15.50	9.2 → 16.85
	Eval	18.29 → 19.65	13.7 → 22.48
DNN	Dev	12.92 → 12.40	11.57 → 13.33
	Eval	15.94 → 17.91	28.95 → 31.46

From the increased EERs of our initial model, it is evident that the effect of this intervention on CMs trained on the original training data containing BCS signatures is much higher in comparison to the new models. We observe that our proposed frame-level DNN model shows the best results under this intervention, demonstrating its robustness on this dataset. Furthermore, under the initial training conditions (without endpoint detection) the error rate of this DNN changes by about 2.5% (on the evaluation set), and is the smallest absolute change among all other initial models (including CNNs). Overall our

proposed approach of training CMs using endpoint detection demonstrates robust performance over the initial models. This holds true for all our CM models studied in this paper.

5.3.4 Discussion

Replay spoofing attack detection, a binary classification problem, in general is a difficult task to solve. As explained in Subsection 2.9.1 confounders or artefacts in a dataset can affect a wide range of machine learning tasks including anti-spoofing systems (see Sections 4.2, 4.6 and Subsection 5.2.5). Such confounders are often overlooked in research studies. We consider two reasons for this. First, the figure of merit used in assessing performance (a scalar) does not account for them and their influence in learning algorithms. Second, they often offer gains in performance (see Subsection 2.9.1 for related background). Due to these reasons we often do not care towards accountability of such ML models trained on data containing artefacts. But, impressive performance reported by such untrustworthy models can be costly as they may fail with high likelihood when used in practical real-world scenarios. Therefore, ensuring reliable performance estimates is important to truly assess the ability of proposed features/classifiers for a given machine learning task.

In this direction, focussing on machine learning tasks for anti-spoofing using the benchmark ASVspoof 2017 v2.0 dataset, this section proposed a method to mitigate the impact of artefacts on this dataset, and build reliable and trustworthy models. For this, a speech endpoint detection module (Fig. 5.1) that discards every audio sample before and after the actual speech utterance was proposed. This ensures that both classes of audio now have a similar pattern, forcing learning algorithms to focus on exploiting factors of interest — for example channel characteristics, in solving the spoofing detection problem, thus producing reliable performance estimates.

As the reliability of the proposed method depends heavily on the accuracy of endpoint annotations, manual annotations were developed and used to train and validate model parameters. During testing, a robust voice activity detection algorithm was used (Subection 5.3.2) to derive endpoint annotations. The correctness of automatic annotations was further verified comparing performance of CM models trained using both automatic and manual annotations (see Fig. 5.2). Next, the proposed method was evaluated on five different countermeasures. This includes CQCC feature based GMM and i-vector based Cosine that were initially used in [Delgado et al., 2018] as baselines for the version 2.0 of this dataset. New benchmark results are provided showcasing the true performance estimates when these confounders are taken into account, making these models

more trustworthy. The section also demonstrated the robustness of the proposed method against being manipulated using signal artefacts. For this, both newly trained countermeasures and the ones trained without endpoint detection were used to assess their robustness. The results confirmed that the proposed method helped mitigate the impact substantially (Subection 5.3.3). Finally, the section also proposed a DNN trained at the frame-level and demonstrated its robustness against artefacts in the dataset. The work described here and part of the work from Section 4.6 is under review in an IEEE journal.

5.4 Subband analysis for spoofing detection

In the previous Sections 4.5 and 5.3, we have seen how deep models are trained on the fullband spectrum of the speech signal using time-frequency representations (spectrograms). While this is a commonly adopted approach in spoofing detection research as evident from the literature in Section 2.8, we argue that not all frequency bands are useful for these tasks. To this end, this section thoroughly investigates the impact of different subbands and their importance on replay spoofing detection. A joint subband modelling framework that employs n sub-networks to learn subband specific features is proposed. These networks are later combined and passed to a classifier and the whole network weights are updated during training. Subsection 5.4.2 provides a detailed explanation of the proposed methodology. Then Subsection 5.4.3 provides details of three different experimental designs. The first design focuses on subband CNNs and a joint subband modelling framework. The second one aims at studying the effect of late fusion of subband CNN models and comparing its performance with the proposed joint model training framework. For this, two linear fusion approaches are investigated: sum of scores and weighted sum of scores where weights are learned through logistic regression. And the final experimental design aims at investigating the generalisability of the proposed subband modelling approaches on the ASVspoof 2019 real PA testset (see Subsection 3.3.3). Then Subsection 5.4.4 evaluates the proposed methods and provides a summary of results for different experimental setups. Finally, Subsection 5.4.5 provides a summary of the work done in this section. The proposed methodology is evaluated on two benchmark datasets: ASVspoof 2017 v2.0 and ASVspoof 2019 PA as described in Section 3.

5.4.1 Introduction

Here, we focus on **feature extraction** for audio spoofing attack detection. There is a vast body of prior research on developing and enhancing different low-level feature extractors (most relevant work is reviewed in Section 2.8), some of them obtaining very low spoof-bonafide detection error rates (even 0%) on specific datasets. Many of these techniques leverage domain knowledge, whether speech science (speech production or perception), signal processing theory, or both. A potential benefit of such rationale is transparency and interpretability. At the same time, feature extractors crafted with the aid of domain knowledge might be too simplistic. As illustrated in Fig. 5.3 (b), we aim at hitting a suitable balance between hand-crafted and data-driven feature extraction: we use **spectrograms** (a meaningful representation of audio), processed in disjoint **subbands** to divide-and-conquer high-dimensional spectrogram modeling

across several, frequency-localized models, each handling a lower-dimensional feature space, each modeled with a **convolutional neural network** to learn band-specific features. The subband-specific features are concatenated to form feature vectors that are then classified with a feedforward neural network.

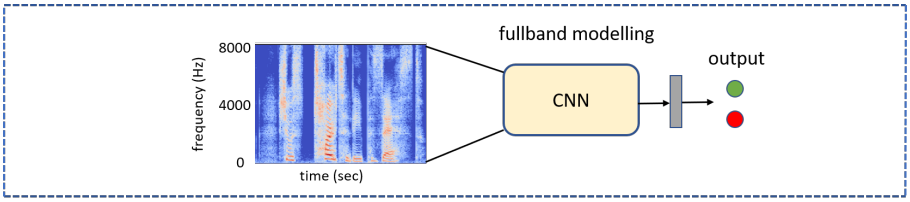
The general idea of processing a power spectrogram in subbands, as such, is not new in the speech field. Mel-frequency cepstral coefficients (MFCCs) are extracted using a filterbank consisting of frequency-localized filters and subband-based modeling of speaker traits dates at least two decades back [Besacier and Bonastre, 2000]. In conventional, or *fullband* models, one trains a single model (with a large number of parameters) using a descriptor of the fullband spectrum. In *subband* based models, the rationale is to instead divide-and-conquer the task across independently modeled subbands which are later recombined using feature or score fusion techniques. The potential benefits include the possibility to side-step the 'curse of dimensionality' by using a set of models trained on lower-dimensional inputs, robustness to *frequency-selective noise*, and the possibility to analyse the importance (contribution) of each subband to the classification results. One potential downside, however, is that the models cannot easily learn or exploit informative correlations between the subbands. Below we summarise the main contributions of this work.

- A systematic study on different subbands and their contribution in replay spoofing detection using convolutional neural networks (CNNs) trained on time-frequency input representations (spectrograms) of the input signal is performed.

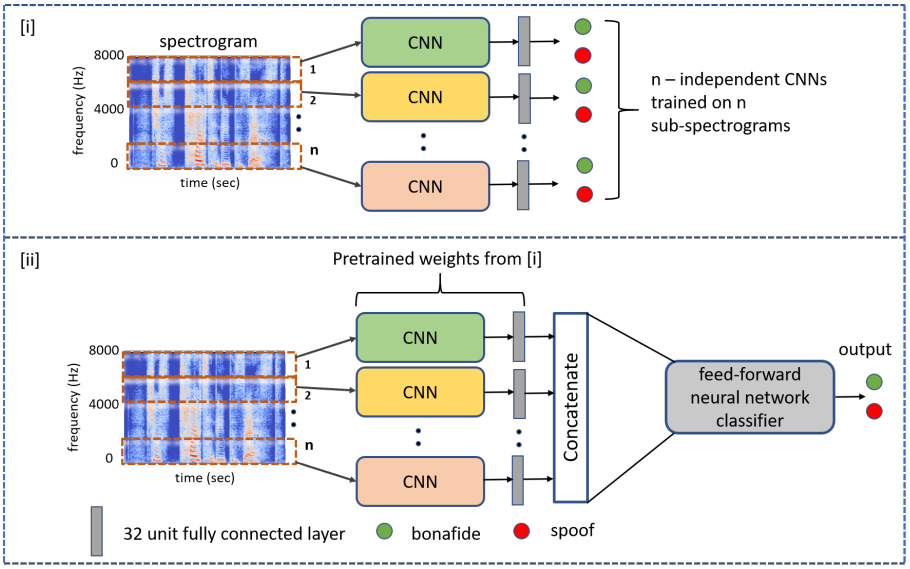
- A joint subband CNN modelling framework is proposed. It works by first splitting the original spectrogram into n sub-spectrograms and training n independent models. The pretrained weights are then used to initialise the weights of the subband joint modelling framework illustrated in Figure 5.3 (b). Subsection 5.4.4 demonstrates the effectiveness of this method offering substantial improvement over traditional fullband modelling.

- We investigate the effect of late fusion of subband CNN models and compare its performance with our proposed joint model training framework. To this end, we investigate simple linear sum fusion and logistic regression based score fusion approaches.

- Finally, we study the generalisability of all our replay spoofing countermeasures on the ASVspoof 2019 "real" PA test set. We perform this study using models trained on both the ASVspoof 2017 and 2019 PA datasets.



(a) Baseline. Traditional CNN model trained on the fullband spectrum.



(b) Proposed framework. (i) The original spectrogram is split into n sub-spectrograms on which n independent CNNs are trained. (ii) uses pretrained weights from (i) to initialise n subband CNNs of the joint subband framework. The fully-connected layers output from n subband models are concatenated and fed to a feed forward neural network for final classification. The whole network weights are updated during training.

Figure 5.3: Proposed subband CNN modeling framework.

5.4.2 Proposed method

Convolutional neural network (CNN) based countermeasure models trained using spectrograms have shown state-of-the-art performance in spoofing detection tasks in the ASVspoof 2017 challenge. They are usually trained using the fullband spectrum of the input signal and use a fixed-duration input representation [Lavrentyeva et al., 2017]. This *conventional approach* of building CNN-based countermeasures is illustrated in Figure 5.3 (a). As the CNN is trained discriminatively, it is forced to learn discriminative features using the entire frequency spectrum of the input signal, using a *single* worker to extract usable information across all the frequency subbands for spoofing attack detection.

But as the prior studies (Section 2.8) suggest, not all the subbands are necessarily equally informative. From a modeling perspective, the raw spectrogram

patch (extracted by stacking multiple frames using all the frequency bands) is a high-dimensional vector, with strong correlations between any neighboring time (frame) or frequency (DFT bin) indices. As such low-level redundancy is common to both human and spoofed samples, it does not necessarily help in the discrimination (classification) task itself; instead, the model will have to learn both data compression (suppressing statistical redundancy to a useful intermediate representation) *and* classification tasks. This may also result in additional computational time during convolution operations.

Therefore, rather than having a single CNN that merges information across different frequency bands, we propose to incorporate a *bank* of n different CNNs, each operating on non-overlapping n frequency subbands. Our proposed methodology is illustrated in Figure 5.3 (b). Each of the *subband CNNs* now has to model a much lower-dimensional subspace, producing a less redundant and more relevant representation of its respective subband. Note that the subband representations are afterwards re-combined through concatenation. This new representation now contains again information across the full frequency band, allowing any subsequent model to exploit possibly useful band-level correlations. In our case, we use a simple feedforward neural network (FFNN) for the final classification. Natural questions that arise now are *how to perform the frequency-domain split* and *how to choose* n. We address three different forms of splits, $n = 2$, $n = 4$ and $n = 8$, and go for the easiest choice of uniform frequency division.

This choice is motivated from [Lin et al., 2018]. They divide the original spectrogram into n uniform subbands with bandwidth 1 kHz corresponding to $n = 8$ splits and 0.5 kHz bandwidth for $n = 16$ splits. They remove one subband at a time and hand-craft features from the remaining subbands. GMMs are then trained on these features for spoofing detection and the performance is evaluated in terms of EER on the ASVspoof 2017 evaluation sets. As our dataset consists of 16 kHz audio (Nyquist range 8 kHz), our three choices correspond to subbands of bandwidths 4 kHz ($n = 2$), 2 kHz ($n = 4$) and 1 kHz ($n = 8$). It should be noted that the default case $n = 1$ corresponds to the baseline CNN (Figure 5.3 a), i.e the model trained on the fullband spectrogram. From hereon we use "CNN" to refer to the baseline CNN. And, we use "sub-CNN" to refer to models trained on the subband spectrograms.

We operate on power spectrograms instead of other alternative time-frequency representations, following findings in [Lavrentyeva et al., 2017]. All our sub-CNNs use the architecture described in Subsection 4.5.2, which is an adapted version of the best performing model [Lavrentyeva et al., 2017] in the ASVspoof 2017 challenge. It consists of 9 convolutional layers, 5 max-pooling layers and 2 fully connected (FC) layers; refer to Subsection 4.5.2 for further details. The key

difference while training such sub-CNNs is in terms of the input they receive. The bandwidth of the input sub-spectrogram varies depending upon different values of n (number of splits). Subsection 5.4.4 provides more details regarding input representations, training and testing of these models.

The proposed joint sub-CNN model of Figure 5.3 (b) second row uses the same architecture as in sub-CNNs (first row of the same figure) with the following updates: (1) there is no output layer now, and (2) a concatenation layer is added that merges the FC layer's output from n sub-CNN models producing a $32 \times n$ dimensional vector. It should be noted that the choice of 32 units in the FC layer comes from the baseline CNN of Figure 5.3 (a). Furthermore, this architecture with 32 FC units has shown promising results as described in Section 4.6. Next, the concatenated vector is fed to a feedforward neural network for class discrimination. The FFNN consists of two fully-connected layers with 256 and 128 units. This is followed by a single unit output layer with sigmoid non linearity for class discrimination. We apply batch normalisation before applying ReLU non-linearity to these layers. The architecture of the FFNN is optimised through model validation (on the development set). Training and optimisation of our proposed framework is done in two steps:

- First, the input spectrogram is split into n non-overlapping sub-spectrograms and n sub-CNNs are trained independently on them. The training dataset is used for model training and the development dataset is used for model validation. This step is depicted in the top row of Figure 5.3(b).

- Second, the pretrained sub-CNNs (excluding the last layer) are used to initialise the weights of the sub-CNN modules of our joint sub-CNN framework shown in the bottom row of Figure 5.3 (b). The weights of the FFNN layers are initialised randomly using xavier initialisation [Glorot and Bengio, 2010]. The biases are initialised to zero. Given an input spectrogram, the framework first splits it into n non-overlapping sub-spectrograms which are processed by n sub-CNNs and the whole network parameters are jointly updated during backpropagation. This step can be interpreted as fine-tuning of the subband CNNs and the classifier back-end jointly for best performance. As in the earlier step, model parameters are trained on the training dataset and the development set is used for model validation.

Our proposed work is different from prior works [Sriskandaraja et al., 2016, Witkowski et al., 2017, Garg et al., 2019, Nagarsheth et al., 2017, Lin et al., 2018] because most of them aim at hand-crafting or learning features [Soni et al., 2016] based on the relevance of specific subbands for spoofing detection.

To the best of our knowledge, there is no work in spoofing detection aiming to learn band-specific features by discriminatively training CNNs on a spectrogram input.

5.4.3 Experimental design

This section describes three different experimental designs. The first experiment discusses various subband and joint subband modelling designs considered in this work. The second experiment describes various fusion setups studied. The third experiment aims at cross-database performance evaluation of subband models on an unseen test set. Finally, a brief summary of baseline experiments is also provided.

Experiment 1: subband modeling. We design four experimental setups for different values of n using our proposed methodology. We use $n = 2$ in our first setup. Using the architecture and training methodology described earlier we train two independent sub-CNNs M_1 and M_2. M_1 operates on the first 4 kHz and M_2 on the last 4 kHz subband spectrograms. Next, we use them (except the last output layer) to initialise the respective sub-CNN module weights of our joint sub-CNN model framework as shown in the bottom row of Figure 5.3(b). We call this joint model as J_1. Our second setup uses $n = 4$. Therefore, we train four independent sub-CNNs M_3 through M_6 on 2 kHz subband spectrograms. We then use these pretrained models to initialise the weights of sub-CNN modules of our joint model framework J_2. Our final setup uses $n = 8$. We now train eight independent sub-CNNs M_7 through M_{14} operating on 1 kHz subband spectrograms. We then use them (except the last layer weights) to initialise our joint model framework J_3 shown in Figure 5.3(b) bottom row.

Finally, motivated from the results of sub-CNNs M_7 and M_{14} on the ASVspoof 2017 dataset (shown in Table 5.13) we design a joint framework model J_4 operating on the first and the last 1 kHz subband spectrograms. This setup is different from the previous setups as more than half of the information is being discarded here. Overall J_4 utilises only 2 kHz of information (first and the last 1 kHz) bands. As in the earlier setups, the pretrained weights (here M_7 and M_{14} models) are used to initialise the weights of sub-CNN modules of our joint model framework J_4. It should be noted that the entire model parameters are jointly optimised while training J_1, J_2, J_3 and J_4. We perform these experiments on both the ASVspoof 2017 v2.0 and 2019 PA datasets.

Experiment 2: score fusion. We perform fusion experiments to understand if combining information through score-level fusion helps improve detection per-

Table 5.9: Fusion system details.

Fusion model ID	Fusion type	Models combined (scores)
F_1	Linear sum	M_1, M_2
F_2	Weighted linear sum	M_1, M_2
F_3	Linear sum	M_3 - M_6
F_4	Weighted linear sum	M_3 - M_6
F_5	Linear sum	M_7 - M_{14}
F_6	Weighted linear sum	M_7 - M_{14}

formance. In this setting we use the scores from each of the pretrained sub-CNNs and combine their scores using two simple approaches: linear sum of scores (LS), and linear weighted sum of scores (WLS). Let $S_1, S_2, S_3, ..., S_n$ represent scores from n sub-CNNs for a test utterance X. The fused score using the two approaches is obtained as: linear sum $= S_1 + S_2 + S_3 + .. + S_n$ and weighted sum $= w_1 * S_1 + w_2 * S_2 + w_3 * S_3 + .. + w_n * S_n$ where $w_1, w_2, w_3, ..., w_n$ are weights corresponding to each sub-CNN score learned using logistic regression (LR). We use the Bosaris toolkit [Brümmer and de Villiers, 2013] for the LR implementation. We perform six different fusion experiments: F_1 - F_6. Table 5.9 summarises the details of the fusion systems.

Experiment 3: cross-database evaluation. The final experimental setup evaluates the performance of our proposed models on an unseen real replay test dataset. In other words, the main goal here is to study the generalisability of our models: M_1 through M_{14} and joint models J_1, J_2, J_3 and J_4 in a *cross-database* evaluation setting. We use all our pretrained models trained on the ASVspoof 2017 and ASVspoof 2019 PA dataset and evaluate their performance on the ASVspoof 2019 real PA test set.

Baseline. To assess the performance of our proposed framework, we train a baseline CNN model on the fullband spectrogram. The model framework is the same as in Figure 5.3(a) except that the number of splits is $n = 1$. For completeness, we also train and test a CQCC-based GMM model so as to compare it with our proposed framework. It should be noted that due to the preprocessing (described in the next section) applied on the ASVspoof 2017 and 2019 PA datasets during training and testing, our baselines are different from the ones reported in [Delgado et al., 2018, Todisco et al., 2019.]; therefore, the numbers reported using our GMM baselines should not be directly compared with the results of ASVspoof challenges. We train GMMs using the training dataset and parameterisation from Delgado et al. [2018].

5.4.4 Evaluation

Datasets

We use two publicly available spoofing datasets: ASVspoof 2017 v2.0 and ASVspoof 2019 physical access (PA) for model training and testing. In addition, we also include results on the recently released ASVspoof2019 real PA dataset[12] for the challenging case of cross-database performance evaluation. All the datasets are representative of *replay attacks* and are complementary to each other. Section 3 provides more details on these datasets. Following our prior findings (Subsection 5.2.5) we adopt a custom, but publicly available protocol[13] described in Subsection 5.2.3 for training models on the ASVspoof 2019 dataset.

Input representation and preprocessing

The input to the network is a mean-variance normalised log power spectrogram of 3 seconds. This normalisation, motivated from [Lavrentyeva et al., 2017], is performed at the utterance-level to standardize the features (zero mean and unit variance for all frequency bins) within a given recording. We use a 512-point *fast Fourier transform* (FFT), and a 32 ms window with a hop of 10 ms. Therefore, the original input spectrogram has a shape of 300×257, where 300 is the number of frames and 257 the number of FFT bins. To obtain a consistent input representation we replicate the audio samples (in the time domain) if the duration is smaller, or truncate the samples to 3 seconds duration. When $n = 1$, the input shape to the baseline CNN remains the same (300×257) however it varies for sub-CNNs depending upon different splits we use. For example, when $n = 2$, the input shape becomes 300×128 and 300×129. We always include the leftover bin to the last split. Similarly, when $n = 4$ we have four sub-spectrograms where the first three will have a shape of 300×64, and a shape of 300×65 for the last split. Likewise, for $n = 8$ we have seven sub-spectrograms of shape 300×32, and a shape of 300×33 for the last split.

Following our prior findings (Subsection 5.2.5) on the ASVspoof 2019 PA dataset, we remove zero-valued samples from the start and end of every audio recording in the dataset. Likewise, on the ASVspoof 2017 v2.0 dataset, we remove leading and trailing silence/nonspeech samples following our findings described in Subsection 4.6.3. For this, we use our publicly released speech endpoint annotations described in Subsection 5.3.2. Applying such preprocessing helps the model avoid exploiting cues that are actually not relevant to the problem, rather this forces the models to now learn relevant factors in replay spoofing detection.

[12] `https://www.asvspoof.org/database`
[13] https://github.com/BhusanChettri/ASVspoof2019

Table 5.10: Performance of the baselines on the ASVspoof 2017 and 2019 evaluation sets. Models are trained and validated on the respective datasets.

Baseline	subbands (kHz)	ASVspoof 2017		ASVspoof 2019	
		t-DCF	EER%	t-DCF	EER%
CNN	0-8	0.3873	13.02	0.2019	7.00
GMM	0-8	0.5054	18.33	0.9928	59.12

Training, testing and evaluation

We train the network to optimise the binary cross entropy loss between a bonafide and a spoof class. We use a batch size of 32 and learning rate of $1e^{-4}$. We use the *ADAM* [Kingma and Ba, 2014] optimiser with default parameters. We apply a dropout of 50% to the input of the fully connected layers. If the validation loss does not improve for 5 epochs we stop the training process to avoid overfitting. We train models for a maximum of 100 training epochs. Using this approach we train 5 models with random initialisation. We choose the model showing the best performance on the development data and use it to test performance on the evaluation data. At test time, for each audio spectrogram we use the model output — the bonafide-class posterior probability — as our detection score. The approach described above is the same for all our models. We use the EER and t-DCF metrics (described in Section 3.5) to evaluate model performance.

Results

We first look at the performance of our baseline models. Table 5.10 summarises this. On the ASVspoof 2017 evaluation set, the GMM model reaches an EER of 18.33% and a t-DCF of 0.5054. As highlighted earlier due to the preprocessing applied during training and testing on this dataset, the EER is worse than the original EER of 13.74% reported by Delgado et al. [2018]. On the ASVspoof 2019 PA[14] evaluation set, EER = 59.12% and t-DCF = 0.9928. The results suggest that GMM models become less confident in making classification decisions when silence cues are removed during training and testing. On both datasets, the CNN baseline outperforms the GMM on both metrics. This demonstrates its effectiveness at learning relevant features useful for discrimination despite the preprocessing applied to the audio signals in contrast to hand-crafted CQCC features used in the GMM.

Now we discuss the performance of our individual sub-CNNs M_1, M_2 and the joint model J_1. Table 5.11 summarises the results. On the ASVspoof 2017

[14]We used the pretrained GMM model from section 5.2.2 to test on the evaluation set with preprocessing of silence. Without preprocessing, the same GMM reports an EER of 10.06% and a t-DCF of 0.1971 on the evaluation set.

Table 5.11: Performance of sub-CNNs M_1, M_2 and joint model J_1. Bold indicates the best performance.

Model	subbands (kHz)	ASVspoof 2017		ASVspoof 2019	
		t-DCF	EER%	t-DCF	EER%
M_1	0-4	0.7045	33.03	0.1925	6.97
M_2	4-8	0.4800	18.95	0.5201	19.69
J_1	-	**0.2893**	**10.63**	**0.1864**	**6.44**

Table 5.12: Performance of sub-CNNs M_3 through M_6 and joint model J_2. Bold indicates the best performance.

Model	subbands (kHz)	ASVspoof 2017		ASVspoof 2019	
		t-DCF	EER%	t-DCF	EER%
M_3	0-2	0.6172	31.20	0.2265	7.97
M_4	2-4	0.9773	41.58	0.4568	17.72
M_5	4-6	0.9995	50.70	0.5903	23.16
M_6	6-8	0.5032	23.50	0.6019	23.59
J_2	-	**0.3343**	**11.78**	**0.1977**	**6.99**

dataset, the higher frequency bands ($4-8$ kHz) seem to carry more discriminative information than the lower bands ($0-4$ kHz). However, we observe the opposite pattern on the ASVspoof 2019 dataset where M_1 trained on $0-4$ kHz shows better results than M_2. This might be due to differences in dataset design (real vs. simulated replay) and compilation (different speakers and audio qualities of the source corpora). Nonetheless, on both the datasets our proposed joint modelling framework J_1 outperforms M_1, M_2 and the baselines by a large margin on both performance metrics. This demonstrates the effectiveness of our proposed approach for spoofing attack detection.

Next we discuss the performance of sub-CNNs M_3 through M_6 and the joint model J_2. Table 5.12 summarises this. Overall, the individual sub-CNNs now show poor performance, which is expected as each of them now receives only half of the information as the previous case ($n = 2$). Nonetheless, the proposed model J_2 that merges information from all the bands again outperforms the fullband baselines (CNN and GMM) and sub-CNNs M_3 through M_6.

Table 5.13 summarises the performance of our individual sub-CNNs M_7 through M_{14} and joint models J_3 and J_4. These results provide deeper insights in understanding the influence of different subbands for spoofing detection. As in the previous two setups ($n = 2$ and $n = 4$), our joint model J_3 shows better results than training individual sub-CNNs indicating that joint training offers some form of complementary information across different frequency bands. We also find that on the ASVspoof 2017 dataset, M_{14} trained on the *last* $7-8$ kHz band outperforms all other sub-CNNs by a large margin. This is followed by

Table 5.13: Performance of sub-CNNs M_7 through M_{14} and joint models J_3 and J_4. Bold indicates the best performance.

Model	subbands (kHz)	ASVspoof 2017		ASVspoof 2019	
		t-DCF	EER%	t-DCF	EER%
M_7	0-1	0.6216	31.59	0.2354	8.48
M_8	1-2	0.9977	48.92	0.6557	25.15
M_9	2-3	0.9971	50.61	0.7327	28.60
M_{10}	3-4	0.9969	49.85	0.5390	21.60
M_{11}	4-5	0.9992	44.06	0.6605	25.69
M_{12}	5-6	0.9914	45.05	0.7118	28.34
M_{13}	6-7	0.9817	42.23	0.7123	28.17
M_{14}	7-8	0.4747	18.10	0.7231	28.01
J_3	-	**0.2734**	11.02	**0.1975**	**7.34**
J_4	-	0.2771	**10.40**	0.2238	8.30

Table 5.14: Score-level fusion results. LS: linear sum of scores. WLS: weighted sum of scores.

Model	Fusion (type)	ASVspoof 2017		ASVspoof 2019	
		t-DCF	EER%	t-DCF	EER%
F_1	LS	0.3208	11.86	0.2208	7.50
F_2	WLS	0.3189	11.72	0.2034	**6.78**
F_3	LS	0.6151	24.25	0.2197	8.00
F_4	WLS	**0.3079**	**11.55**	0.1898	6.95
F_5	LS	0.4932	18.25	0.2626	9.78
F_6	WLS	0.3548	12.33	**0.1848**	6.99

M_7 (operating on the $0-1$ kHz band) that also performs substantially better than the remaining sub-CNNs. Inspired by this finding, we further train another joint model J_4 trained *only* with the lowest and highest sub-CNNs. Interestingly, this highly reduced model yields the best performance on the ASVspoof 2017 dataset, matching with the findings reported on the version 1.0 dataset by Lin et al. [2018]. However, on the ASVspoof 2019 dataset, the first 1 kHz subband appears to carry most relevant information as opposed to other subbands. Furthermore, the results found for the ASVspoof 2017 and 2019 dataset do not match completely with the main differences being in dataset design and collection — the two datasets ASVspoof 2017 and 2019 PA are designed and collected differently.

Next we discuss the performance of our fusion models. Table 5.14 shows these results. In general, WLS fusion shows better performance than LS fusion, outperforming the two baselines (Table 5.10). However they show poor (or similar) performance compared to our joint subband modeling framework. For example, on the ASVspoof 2017 evaluation set the joint model J_4 (Table 5.13) shows better results in comparison to all our fusion models. Similarly, on

Table 5.15: Cross dataset performance evaluation on the unseen ASVspoof 2019 real PA testset. D1: ASVspoof 2017 v2.0, D2: ASVspoof 2019 PA. Highlighted in bold indicates the best performing subband models.

Model	subbands	Trained on D1		Trained on D2	
	(kHz)	t-DCF	EER%	t-DCF	EER%
CNN	0-8	0.9986	41.29	0.6374	34.25
M_1	0-4	0.8205	44.81	0.6800	33.12
M_2	4-8	0.9687	42.59	0.7820	39.44
J_1	-	0.9939	44.07	0.6741	35.92
M_3	0-2	0.8984	38.33	0.6760	33.49
M_4	2-4	**0.8025**	46.08	0.7311	34.62
M_5	4-6	0.9995	59.44	0.8033	35.74
M_6	6-8	0.9976	51.85	0.6877	28.37
J_2	-	0.9944	40.55	0.6544	30.18
M_7	0-1	0.8508	**34.81**	0.6678	34.95
M_8	1-2	0.9875	47.63	1.0	51.29
M_9	2-3	0.9545	51.48	0.8014	32.22
M_{10}	3-4	0.9634	42.22	0.8553	35.41
M_{11}	4-5	0.9003	43.33	0.7524	32.80
M_{12}	5-6	0.9788	47.03	0.8322	38.65
M_{13}	6-7	0.9828	48.49	0.7038	28.70
M_{14}	7-8	0.9981	42.38	**0.6483**	**27.22**
J_3	-	0.9208	37.61	0.6439	30.97
J_4	-	0.9081	36.62	0.7541	36.71

the ASVspoof 2019 PA evaluation set the joint model J_1 (Table 5.11) shows better EER (and slightly worse t-DCF). As expected, combining information through score-fusion techniques offers gain in detection performance. However, our proposed joint subband modelling framework shows better results over score-fusion approaches. This further confirms that the complementary information provided by individual sub-CNNs helps improve overall detection performance.

Table 5.15 summarises the results of cross-database evaluation. Our models trained on the ASVspoof 2017 v2.0 and ASVspoof 2019 PA datasets show poor performance on the ASVspoof 2019 real PA test set. Though they show good performance on the respective evaluation sets (see Tables 5.11, 5.12, 5.13), they do not generalise well to unseen replay attack conditions. On both the t-DCF and EER metrics, we observe high error rates for our joint models J_1, J_2, J_3 and J_4. Furthermore, most of our individual sub-CNN models M_1 through M_{14} show poor generalisation. This indicates overfitting on the respective datasets and lack of generalisability in unseen attack conditions. One possible interpretation to this observation may be attributed to dataset design and collection for the ASVspoof 2017 and 2019 PA datasets. It is worth noting that the ASVspoof 2019 PA dataset is developed through controlled simulation while the ASVspoof

Table 5.16: Summary of results showing the comparison of baselines with our proposed models. * results taken from Table 5.10.

Model	ASVspoof 2017		ASVspoof 2019	
	t-DCF	EER%	t-DCF	EER%
GMM*	0.5054	18.33	0.9928	59.12
CNN*	0.3873	13.02	0.2019	7.0
J_1	0.2893	10.63	**0.1864**	**6.44**
J_2	0.3343	11.78	0.1977	6.99
J_3	**0.2734**	11.02	0.1975	7.34
J_4	0.2771	**10.40**	0.2238	8.30

2017 dataset is collected in real world recording and replay conditions. Due to the dataset issues identified on both the ASVspoof 2017 and 2019 PA datasets, the models trained on these datasets might not be able to capture real replay attack conditions and thus perform poorly on the real PA test set which has been designed carefully reflecting real replay attack conditions. This study further suggests that there is still need for a reliable replay training dataset that can be used to train models incorporating real world replay attack conditions.

5.4.5 Discussion

This section presented a detailed analysis on the impact of different subbands and their importance on replay spoofing detection tasks on the benchmark datasets ASVspoof 2017 v2.0 and ASVspoof 2019 PA. Our proposed subband CNN model outperformed the traditional fullband CNN model, and also the CQCC-GMM baselines by a large margin, demonstrating the significance of our approach. We also investigated how combining information at the score level from each subband CNNs compared with our joint subband modelling framework. The final set of studies we performed here is on cross-database evaluation to investigate the generalisability of replay spoofing countermeasures on the ASVspoof 2019 real PA dataset. Table 5.16 provides a summary of our main findings on both the ASVspoof 2017 and 2019 PA datasets. Performance improvements obtained using our proposed approach over baselines are very promising, encouraging further research on subband modelling for spoofing detection. Furthermore, it should be noted that our main objective in this section is not to beat the best performing models published on the two datasets, but to validate our hypothesis about subband modelling using CNNs trained on spectrogram inputs.

Our findings on the ASVspoof 2017 dataset suggest that the most discriminative information appears to be in the first and the last 1 kHz frequency bands, and the joint model trained on these two subbands shows the best performance

outperforming the baselines by a large margin. However, these findings do not generalise on the ASVspoof 2019 PA dataset. Likewise, we find similar observations when we tested our models on the ASVspoof 2019 real replay test recordings. Furthermore, the poor cross-database experimental results suggest that we still have some gaps to fill towards building datasets to study replay spoofing detection for real world conditions.

5.5 Deep VAEs for spoofing detection

So far, this thesis has investigated various machine learning and deep learning approaches for spoofing attack detection in the form of discriminative models. While Chapter 4 has mainly focussed on the analysis of existing methods, this chapter focusses on novel approaches for spoofing detection. Deep models, usually CNNs, have been widely used for discriminative feature learning and generative models (GMMs) have been trained on them for class discrimination showing promising results [Lavrentyeva et al., 2017, Sriskandaraja et al., 2018]. Motivated from the widespread use of GMMs, a generative model, for audio spoofing detection, this section explores the potential of VAEs, a deep generative model for spoofing detection. An introduction and the motivation of this work is first provided in Subsection 5.5.1. Detailed background on GMMs and VAEs is provided in Subsections 2.7.3 and 2.7.1 respectively. Additional details relating GMMs and VAEs as latent variable models are provided in Subsection 2.7.3.

Then Subsection 5.5.2 describes our proposed methods of using VAEs under two settings. First, we explore the potential of VAEs as a backend classifier. For this, three different approaches to condition VAE training are investigated. The second setting is on using the VAE as a feature extractor. For this, the VAE residual, which is the absolute difference of the original input and the reconstruction, is proposed as a new feature for spoofing detection. The next Subsection 5.5.3 provides a description of datasets used, features and input representations considered, model architectures, training and testing approaches, and evaluation metrics. Then Subsection 5.5.4 evaluates our proposed methods and provides a summary of different experimental results. It first evaluates the impact of latent space dimensionality on the VAE performance. Then, a performance comparison of different VAE setups with the baseline GMMs is provided. Next, this section evaluates the performance of VAEs trained with multi-class conditioning and compares them with VAEs trained using binary-class conditioning. Then, a qualitative analysis through t-SNE visualisations is provided to gain insights on the latent space learned by the VAE. Next, this section evaluates the performance of our proposed VAE residual features. Finally, Subsection 5.5.5 provides a summary of the work done in this section. The work reported here was published in [Chettri et al., 2020].

5.5.1 Introduction

In classic automatic speech recognition (ASR) systems and many other speech applications, prior knowledge of speech acoustics and speech perception has guided the design of some successful feature extraction techniques, *mel frequency*

cepstral coefficients (MFCCs) [Davis and Mermelstein, 1980] being a representative example. Similar *a priori* characterization of acoustic cues that are relevant for spoofing attack detection, however, is challenging; this is because many attacks are *unseen*, and since the human auditory system has its limits — it is not designed to detect spoofed speech and may therefore be a poor guide in feature crafting. This motivates the study of data-driven approaches that learn automatically relevant representations for spoofing detection. Both discriminative models (such as *support vector machines* (SVMs), *deep neural networks* (DNNs) [Nagarsheth et al., 2017, Zhang et al., 2017]) and generative models (such as GMMs) [Patel and Patil, 2015, Lavrentyeva et al., 2017], have extensively been used as backends for spoofing detection. The former directly optimise the class decision boundary while the latter model the data generation process within each of the classes, with the decision boundary being implied indirectly. Both approaches can be used for classification but only the generative approach can be used to sample new data points. We focus on generative modeling as it allows us to interpret the generated samples to gain insights about our modeling problem, or to "debug" the deep learning models and illustrate what the model has learned from the data to make decisions. Further, they can be used for data augmentation which is challenging using purely discriminative approaches. Subsections 2.7.3 and 2.7.1 provide necessary background on GMMs and VAEs respectively. Furthermore, Subsection 2.7.3 provides additional details relating them as latent variable models.

GMMs have empirically been demonstrated to be competitive in both the ASVspoof 2015 and ASVspoof 2017 challenges [Patel and Patil, 2015, Lavrentyeva et al., 2017]. While Patel and Patil [2015] used hand-crafted features for synthetic speech detection, Lavrentyeva et al. [2017] used deep features to train GMM backends. A known problem with GMMs, however, is that the use of high-dimensional (short-term) features often leads to numerical problems due to singular covariance matrices. Even if off-the-shelf dimensionality reduction methods such as *principal component analysis* (PCA) [Jolliffe and Cadima, 2016] or *linear discriminant analysis* (LDA) [Tharwat et al., 2017] prior to GMM modeling may help, they are not jointly optimised with the GMM. Is there an alternative way to learn a generative model that can handle high-dimensional inputs natively? Three generative models that have been demonstrated to produce excellent results in different applications include *generative adversarial networks* (GANs) [Goodfellow et al., 2014], *variational autoencoders* (VAEs) [Kingma and Welling, 2013] and *autoregressive* models (for example, *WaveNet* [Oord et al., 2016]). Both GANs and VAEs have demonstrated promising results in computer vision [Pu et al., 2016, Gulrajani et al., 2016, Walker et al., 2016], video generation [Tulyakov et al., 2017] and natural language processing tasks

[Subramanian et al., 2017]. VAEs have recently been studied for modeling and generation of speech signals [Blaauw and Bonada, 2016, Hsu et al., 2017b,a], and synthesizing music sounds in [Esling et al., 2018]. They have also been used for speech enhancement [Leglaive et al., 2019, Kameoka et al., 2018b] and feature learning for ASR [Tan and Sim, 2016, Feng and Lee, 2019]. Recent studies in ASV have studied the use of VAEs in *data augmentation* [Wu et al., 2019], *regularisation* [Zhang et al., 2019b] and *domain adaptation* [Tu et al., 2019] for deep speaker embeddings (x-vectors). In TTS, VAEs have been recently used to learn speaking style in an end-to-end setting [Zhang et al., 2019a]. Recent work in [Yang et al., 2019b] uses VAEs for extracting low-dimensional features and trains a separate classifier on these features for spoofing detection.

However, to the best of the author's knowledge, the application of VAEs as a backend classifier for spoofing attack detection in ASV remains an unexplored avenue. This section explores deep probabilistic VAEs under two settings. First, as a backend for spoofing detection. Fig. 5.4 illustrates this idea. Second, as a feature extractor (Fig. 5.6) where we propose VAE residuals — the absolute difference of the original input and the reconstruction — as a new feature representation. We are motivated to consider VAEs among other generative models (GANs, WaveNets) because they have both the inference and generator networks, and are more naturally suited to our tasks. The reconstruction quality of VAEs tends to be inferior to that obtained by GANs [Huang et al., 2018], but for classification tasks, obtaining a perfect reconstruction is *not* the main priority. A key challenge, instead, is how to train VAEs to not only preserve reasonable reconstruction but to allow to retain discriminative information in the latent space. To address this, VAEs are often trained with additional constraints. For example, by conditioning the encoder and decoder with additional inputs — so called *conditional* VAEs (C-VAEs) [Kihyuk Sohn and Honglak Lee and Xinchen Yan, 2015]; or by augmenting an *auxiliary classifier* either to the latent space [Li et al., 2019a] or to the output of the decoder network [Kameoka et al., 2018a]. As there is no *de facto* standard for this, we aim to fill this knowledge gap in the domain of audio replay detection. We summarise the contributions of this work as follows:

- While deep generative models, VAEs in particular, have been studied in many other domains, their application in audio spoofing detection remains less explored to date. We study the potential of deep generative VAEs as a backend classifier for spoofing detection. To the best of our knowledge, this is the first work in this direction.

- We describe practical challenges in training a VAE model for spoofing detection applications and discuss approaches that can help overcome them,

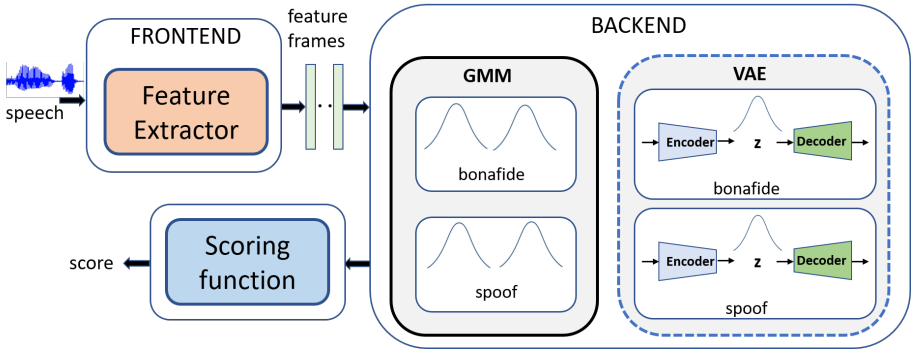


Figure 5.4: Spoofing countermeasure pipeline using a generative model backend.

which could serve as potential guidelines for others.

- Along with a "naive"[15] VAE we also study conditional VAEs (C-VAEs) [Kihyuk Sohn and Honglak Lee and Xinchen Yan, 2015]. The C-VAE uses class labels as an additional conditional input during training and inference. Since we pass class labels in the C-VAE, we use a single model to represent both classes unlike the naive VAE where we train two separate models, one each for bonafide and spoof class. For the text-dependent setting of ASVspoof 2017 data, we further address conditioning using a combination of the class and passphrase labels.

- Inspired by [Li et al., 2019a, Kameoka et al., 2018a], we introduce an *auxiliary classifier* into our VAE modeling framework and study how this helps the latent space[16] to capture discriminative information without sacrificing much during reconstruction. We experiment adding the classifier model on top of the latent space and at the end of the decoder.

- While our primary focus is in using VAEs as a back-end, we also address their potential in feature extraction. In particular, we subtract a VAE-modeled spectrogram from the original spectrogram so as to de-emphasize the importance of salient speech features (hypothesized to be less relevant in spoofing attack detection) and focus on the residual details. We train a convolutional neural network classifier using these residual features.

[15] We use *naive* VAE to refer the standard (vanilla) VAE [Kingma and Welling, 2013] trained without any class labels. Information about the class is included by independently training one VAE per class.

[16] A latent space is a probability distribution that defines the observed-data generation process and is characterised by means and variances of the encoder network.

5.5.2 Proposed method

This section now provides a detailed description of our two proposed methods for spoofing detection. The first method explores the potential of VAEs as a classifier for spoofing detection. For this, three different VAE settings are studied. Then the second method explores VAEs as a feature extractor. For this, VAE residuals which is the absolute difference of the original input and model reconstruction is proposed as a robust feature that retains the factors of interest for replay spoofing detection. A separate classifier is then trained on these features for final classification.

VAE as an alternative backend classifier

As described in Subsection 2.7.3, the VAE is an unsupervised method that learns an encoder-decoder pair, $\boldsymbol{\Lambda} = (\boldsymbol{\theta}, \boldsymbol{\phi})$, without requiring class labels. When used for *classification*, rather than data reconstruction, we have to condition VAE training with the class label. Here, we use labels $y_n = 1$ (bonafide) and $y_n = 0$ (spoof) to indicate whether or not the n^{th} training exemplar represents bonafide speech[17]. We consider three alternative approaches to condition VAE training, described as follows.

The first, **naive** approach, is to train VAEs similarly as GMMs [Patil et al., 2017, M S and Murthy, 2018, Todisco et al., 2017] — independently of each other, using the respective training sets $\mathcal{X}_{\text{bona}} = \{\boldsymbol{x}_n | y_n = 1\}$ and $\mathcal{X}_{\text{spoof}} = \{\boldsymbol{x}_n | y_n = 0\}$. VAEs are trained to optimise the loss function described in (2.11). This yields two VAEs, $\boldsymbol{\Lambda}_{\text{bona}}$ and $\boldsymbol{\Lambda}_{\text{spoof}}$. At test time, they are independently scored using (2.11), and combined by subtracting the spoof score from the bonafide score. The higher the score, the higher the confidence that the test utterance originates from the bonafide class.

Our second approach is to train a single **conditional VAE** (C-VAE) [Kihyuk Sohn and Honglak Lee and Xinchen Yan, 2015] model. In contrast to the naive approach, the C-VAE can learn more complex (*e.g.*, multimodal) distributions by including auxiliary inputs (conditioning variables) to the encoder and/or decoder distributions. In this approach, the label vector $\boldsymbol{y}_n$ is used both in training and scoring. Specifically, in our implementation inspired from [Dahmani et al., 2019, Wu et al., 2019], we augment $\boldsymbol{y}_n$ to the output of the last convolutional layer in the encoder network and to the input of the decoder network. Subsection 5.5.3 describes our encoder and decoder architectures. The

[17] We use the vector notation $\boldsymbol{y}_n$ to indicate the corresponding *one-hot vector* — *i.e.*, $\boldsymbol{y}_n = (0, 1)$ to represent bonafide and $\boldsymbol{y}_n = (1, 0)$ to represent a spoof sample.

training loss (2.11) is now revised as:

$$\ell_n(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n, \boldsymbol{y}_n)}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n|\boldsymbol{z}, \boldsymbol{y}_n)\Big] + \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n, \boldsymbol{y}_n) \,\|\, p(\boldsymbol{z}|\boldsymbol{y}_n)\big), \tag{5.1}$$

where, in practice, we relax the class-conditional prior distribution of the latent variable to be independent of the class, *i.e.* $p(\boldsymbol{z}|\boldsymbol{y}_n) = p(\boldsymbol{z})$ [Kihyuk Sohn and Honglak Lee and Xinchen Yan, 2015]. We perform scoring in the same way as for the previous approach: we pass each test exemplar $\boldsymbol{x}$ through the C-VAE using genuine and spoof class vectors $\boldsymbol{y}_n$, to give two different scores, which are then differenced as before. Note that $\boldsymbol{y}_n$ may include any other available useful metadata besides the binary bonafide/spoof class label. In our experiments on the *text-dependent* ASVspoof 2017 corpus consisting of 10 fixed passphrases, we will address the use of class labels and phrase identifiers jointly.

Our third approach is to use an **auxiliary classifier with a conditional VAE** (AC-VAE) to train a discriminative latent space. We use $r_{\boldsymbol{\psi}}(\boldsymbol{x})$ to denote the predicted posterior probability of the bonafide class, as given by an auxiliary classifier (AC); $\boldsymbol{\psi}$ denotes the parameters of AC. Note that the posterior for the spoof class is $1 - r_{\boldsymbol{\psi}}(\boldsymbol{x})$ as there are two classes. Inspired by [Tu et al., 2019] and [Kameoka et al., 2018a], we consider two different AC setups. First, following [Tu et al., 2019], we use the mean $\boldsymbol{\mu}_{\boldsymbol{z}}$ as the input to an AC which is a feedforward neural network with a single hidden layer. Second, following [Kameoka et al., 2018a], we augment a deep-CNN as an AC to the output of the decoder network. Here, we use the CNN architecture from Subsection 4.5.2. From hereon, we call these two setups as AC-VAE$_1$ and AC-VAE$_2$ respectively. To train the model, we jointly optimise the C-VAE loss (5.1) and the AC loss. In specific, the loss for the n^{th} training exemplar is:

$$\ell_n(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \alpha \cdot \ell_n(\boldsymbol{\theta}, \boldsymbol{\phi}) + \beta \cdot \ell_n(\boldsymbol{\psi}), \tag{5.2}$$

where the non-negative control parameters α and β weigh the relative importance of the regularisation terms during training, set by cross-validation[18], and where $\ell_n(\boldsymbol{\psi})$ denotes the binary *cross-entropy loss* for the training exemplar $\boldsymbol{x}_n$. It is defined as:

$$\ell_n(\boldsymbol{\psi}) = -\big(y_n \,\log\, r_{\boldsymbol{\psi}}(\boldsymbol{x}_n) + (1 - y_n)\,\log(1 - r_{\boldsymbol{\psi}}(\boldsymbol{x}_n))\big) \tag{5.3}$$

Note that setting $\alpha = 1$ and $\beta = 0$ in (5.2) gives (5.1) as a special case. At test time we discard the auxiliary classifier and follow the same approach for scoring as in the C-VAE setup discussed earlier. All the three approaches are

[18] We use 0.5 for both α and β.

summarised in Fig. 5.5.

VAE as a feature extractor — VAE residuals

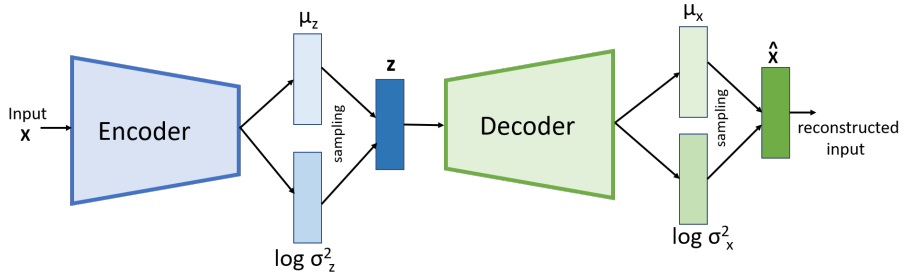
The results shown in Fig. 5.7 indicate that our VAEs have learnt to reconstruct spectrograms using prominent acoustic cues and, further, the latent codes visualised in Fig. 5.8 indicate strong content dependency. The latent space in a VAE may therefore focus on retaining information such as broad spectral structure and formants that help in increasing the data likelihood leading to good reconstruction. But in spoofing attack detection (especially the case of high-quality replay attacks) we are also interested in *detail* — the part *not* modeled by a VAE. This motivates us to consider an alternative use case of the VAE as a *feature extractor*. The idea is illustrated in Fig. 5.6. We use a pre-trained C-VAE model (with bonafide-class conditioning) to obtain a new feature representation that we dub as **VAE residual**, defined as the absolute difference of the input spectrogram and the reconstructed spectrogram by the C-VAE model. We extract the VAE residual features from all training utterances and train a new classifier backend (here, a CNN) using these features as input. We adopt the CNN architecture and training from Subsection 4.5.2. During testing, we use the CNN output activation (Sigmoid activation) as our spoof detection score. Though another recent study also used VAEs for feature extraction [Yang et al., 2019b], our approach is different; they used the latent variable from a pretrained VAE model, while we use the residual of the original and reconstructed inputs.

5.5.3 Experimental setup

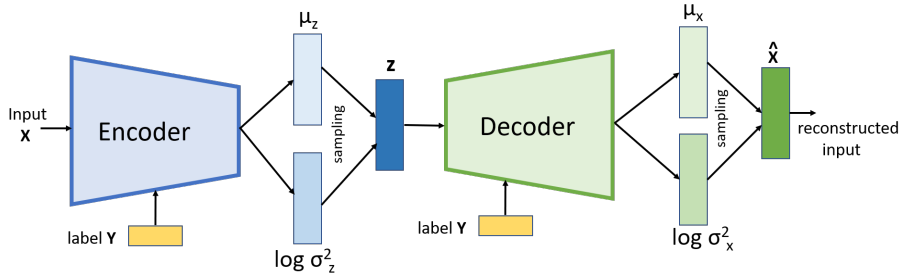
We now describe our experimental setup. This includes description of the evaluation datasets, inputs and feature representation, model architectures, training and testing methods. A brief summary of the metrics considered for performance evaluation is also provided.

Datasets

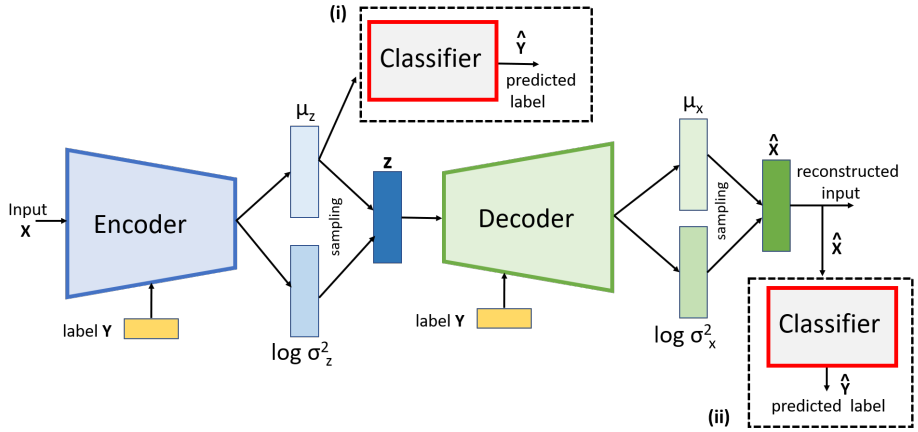
We use two publicly available spoofing datasets: ASVspoof 2017 v2.0 and ASVspoof 2019 physical access (PA) for model training and testing. Section 3 provides more details on these datasets. Furthermore, following our prior findings in Section 5.2, we adopt a custom protocol (described in Subsection 5.2.3) for training and validating our models on the ASVspoof 2019 dataset.



(a) **Naive VAE**. Separate bonafide and spoof VAE models are trained using the respective-class training audio files.



(b) **C-VAE**. A single VAE model is trained using the entire training examples but with class-label vectors.



(c) **AC-VAE** extends **C-VAE** by augmenting an auxiliary classifier (AC). We include AC in two alternative settings: (i) AC-VAE$_1$: use latent mean vector $\boldsymbol{\mu_z}$ as its input, or (ii) AC-VAE$_2$: at the end of decoder using reconstruction as its input. These are highlighted with dotted lines. At test time we discard the AC.

Figure 5.5: Different VAE setups investigated.

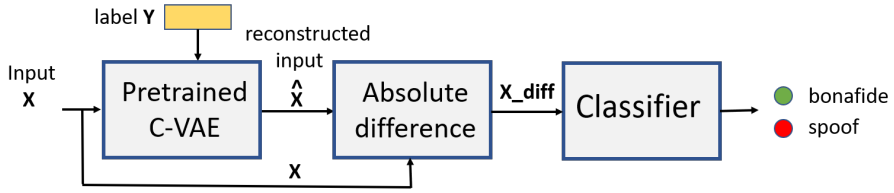


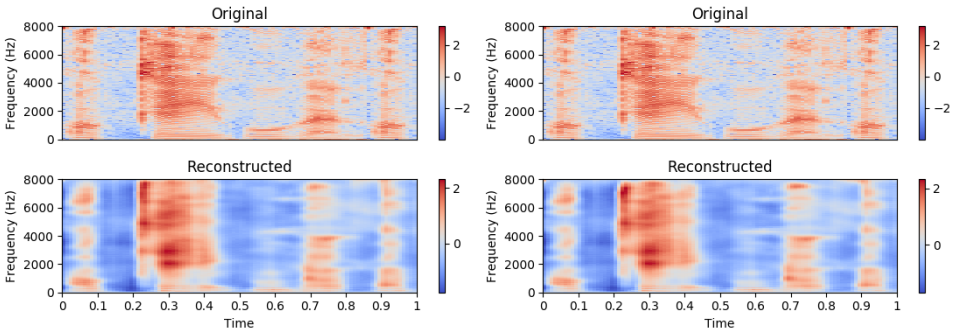
Figure 5.6: Proposed countermeasure design using **VAE residual** — the difference of the original and reconstruction.

Features and input representations

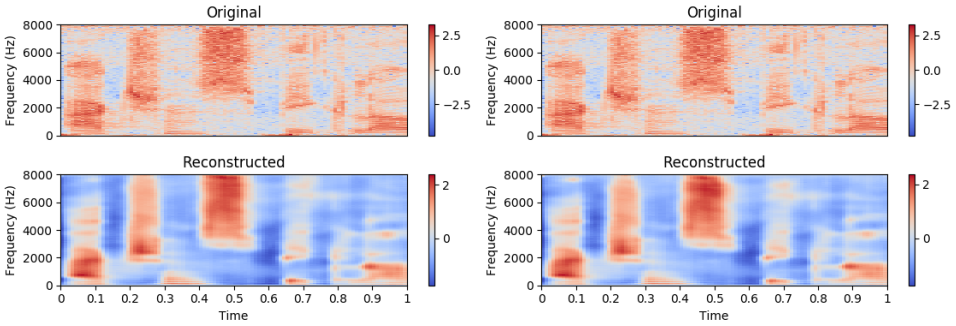
We consider both CQCC [Todisco et al., 2017] and log-power spectrogram features. On the ASVspoof 2017 v2.0 dataset, following our findings in Subsection 4.6.3, we trim silence/noise before and after the utterance in the training, development and test sets. Similarly, on the ASVspoof 2019 PA dataset, following our findings in Subsection 5.2.5 we remove silence from the start and end of the utterances in the whole dataset. We apply these pre-processing steps to ensure that models do not exploit dataset artefacts that bias their decisions (see Subsections 4.6.3 and 5.2.5). Following [Delgado et al., 2018], we extract log energy plus 19-dimensional CQCC static coefficients augmented with deltas and double-deltas, yielding 60-dimensional feature vectors per frame. This is followed by cepstral mean and variance normalisation. As for the power spectrogram, we use a 512-point *discrete Fourier transform* (DFT) with frame size and shift of 32 ms and 10 ms, respectively, leading to N feature frames with 257 spectral bins.

As our VAE models use a fixed input representation, we create a unified feature matrix by truncating or replicating the feature frames. If N is less than our desired number of feature frames T, we copy the original N frames from the beginning until the desired T frames are obtained. Otherwise, if $N > T$, we retain the first T frames. The point of truncating (or replicating) frames in the way described above is to ensure meaningful comparison where both models use the same audio frames as their input. This also means that the numbers reported in this paper are not directly[19] comparable to those reported in literature; in specific, excluding the trailing audio (mostly silence or nonspeech) after the first T seconds will increase the error rates of our baseline GMM substantially. The issue with the original, 'low' error rates relates in part to database design issues, rather than bonafide/spoof discrimination (see Subsections 5.2.5 and 4.6.3). The main motivation to use the T frames at the beginning is to build fixed-length utterance-level countermeasure models, which is a commonly adopted design

[19]GMMs reported in the literature do not truncate or replicate, and this was done by us for a fair comparison with VAEs.



(a) Bonafide file with bonafide-class conditioning (b) Bonafide file with spoof-class conditioning



(c) Spoof file with bonafide-class conditioning (d) Spoof file with spoof-class conditioning

Figure 5.7: Visualisation of the reconstructed spectrograms by the C-VAE. Bonafide audio file (D_1000022.wav) and spoof audio file (D_1001049.wav) are taken from the ASVspoof 2017 v2.0 development set.

Table 5.17: Encoder architecture. Conv: convolutional. T, F: the number of time frames and feature dimensions. The scalar f is the dimension of Conv_5 output flattened vector. M is 16 and 32 for spectrogram and CQCC inputs. Conv_5 is not applicable for CQCCs.

Layer	**Input shape**	**Filter size**	**Stride size**	**# Filters/ neurons**	**Output shape**
Conv_1	T×F×1	5×257	2×2	M	T/2×F/2×M
Conv_2	T/2×F/2×M	5×129	2×2	2M	T/4×F/4×2M
Conv_3	T/4×F/4×2M	5×65	2×2	4M	T/8×F/8×4M
Conv_4	T/8×F/8×4M	5×33	2×2	8M	T/16×F/16×8M
Conv_5	T/16×F/16×8M	5×17	2×2	16M	T/32×F/32 ×16M
$\boldsymbol{\mu_z}$	f	-	-	128	128
$\log \boldsymbol{\sigma_z^2}$	f	-	-	128	128

Table 5.18: Decoder architecture. ConvT: transposed convolutional. * denotes zero padding to match the input shape. The Gaussian layers $\boldsymbol{\mu_x}$ and $\log \boldsymbol{\sigma_x^2}$ use Conv layer. H and W values depends on the number of neurons (#neurons) in the FC layer which is 12288 and 2304 for spectrogram and CQCC inputs, respectively.

Layer	**Input shape**	**Filter size**	**Stride size**	**# Filters/ neurons**	**Output shape**
FC	128	-	-	#neurons	#neurons
ConvT	H×W×128	5×10	2×2	64	2H×2W×64
ConvT	2H×2W×64	5×20	2×2	32	4H×4W×32
ConvT*	5H×4W×32	5×20	2×2	16	10H×8W×16
ConvT*	10H×8W×16	5×20	2×2	8	20H×16W×8
$\boldsymbol{\mu_x}$*	100×F×8	5×5	1×1	1	100×F×1
$\log \boldsymbol{\sigma_x^2}$*	100×F×8	5×5	1×1	1	100×F×1

choice for anti-spoofing systems, *e.g.* [Lavrentyeva et al., 2017, Zhang et al., 2017].

This yields a 100×60-dimensional CQCC representation and a 100×257 power spectrogram representation for every audio file. We use the same number of frames ($T = 100$) for both the GMM and VAE models. Note that GMMs treat frames as independent observations while VAEs consider the whole matrix as a single high-dimensional data point.

Model architecture

Our baseline GMM consists of 512 mixture components (motivated from [Delgado et al., 2018]) with diagonal covariance matrices. As for the VAE, our encoder and decoder model architecture is adopted from [Mishra et al., 2019]. For a given $T \times D$ feature matrix, where T=time frames and D=feature dimension, the encoder predicts the mean $\boldsymbol{\mu}_z$ and the log-variance $\log \boldsymbol{\sigma_z^2}$ that

parameterise the posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, by applying a series of *strided 2D convolutions* [Dumoulin and Visin, 2016] as detailed in Table 5.17. We use a stride of 2 instead of pooling for downsampling the original input. The decoder network architecture is summarised in Table 5.18. It takes a d-dimensional sampled $\boldsymbol{z}$ vector as input and predicts the mean $\boldsymbol{\mu}_{\boldsymbol{x}}$ and the log-variance $\log \boldsymbol{\sigma}^2_{\boldsymbol{x}}$ that parameterise the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ through a series of *transposed convolution* [Dumoulin and Visin, 2016] operations. We use LeakyReLU [Maas et al., 2013] activations in all layers except the Gaussian mean and log variance layers which use linear activations. We use batch normalisation before applying the non-linearity in both the encoder and decoder networks.

Training, scoring and performance metrics

We train GMMs for a maximum of 100 EM iterations with random initialisation of parameters. We train bonafide and spoof GMMs separately to model the respective class distributions. We use only the training partition to train GMMs for both the ASVspoof 2017 and 2019 datasets. At test time, the score of each test utterance is computed as the log likelihood ratio between the bonafide and spoofed GMM model as described in Equation 4.1. We train our VAE models using stochastic gradient descent with Adam optimisation [Kingma and Ba, 2014], with an initial learning rate of 10^{-4} and 16 samples as the minibatch size. We train them for 300 epochs and stop the training if the validation loss does not improve for 10 epochs. We apply 50% dropout to the inputs of fully connected layers in our auxiliary classifier. We do not apply dropout in the encoder and decoder networks.

As for the performance evaluation of our models we use the EER and t-DCF metrics described in Subsections 3.5.1 and 3.5.2 respectively.

Experiments

We perform several experiments using different VAE setups (described in Section 5.5.2) using CQCCs and log-power spectrogram inputs. We also train baseline GMMs for comparing VAE performance using the same input CQCC features. While training VAEs with an auxiliary classifier on the $\boldsymbol{\mu}_{\boldsymbol{z}}$ input, we use 32 units in the FC layer. We do not use the entire training and development audio files for training and model validation on the ASVspoof 2019 dataset, but adopt our proposed protocols described in Subsection 5.2.3. We use them here as they showed good generalisation on the ASVspoof 2019 test dataset during the recent ASVspoof 2019 evaluations (Subsection 5.2.4). Note, however, that all the evaluation portion results are reported on the standard ASVspoof protocols.

Table 5.19: Showing the effect of latent dimensions on the performance metric for the C-VAE model when it is trained on CQCC and spectrogram inputs.

	Spectrogram		CQCC	
Latent dimension	EER%	t-DCF	EER%	t-DCF
8	31.20	0.8642	33.35	0.8584
16	26.88	0.7551	33.74	0.8542
32	36.65	0.9383	30.81	0.7909
64	29.73	0.7650	29.52	0.7325
128	29.43	0.7303	29.27	0.7222
256	29.80	0.7609	28.87	0.6962
512	25.73	0.6662	28.42	0.7033

5.5.4 Evaluation

This section evaluates our proposed methods providing a summary of results for different experimental setups. First, it evaluates the impact of latent space dimensionality on the VAE performance and finds 128 as the optimal choice. This dimension is then used in all our VAE setups. Then, a performance comparison between different VAEs with the baseline GMMs is provided. Next, this section evaluates the performance of VAEs trained with multi-class conditioning and contrasts it with the ones trained using binary-class conditioning. Then, a qualitative analysis through t-SNE visualisations is provided to gain insights on the latent space learned by the VAE. Finally, this section evaluates the performance of our proposed VAE residual features by training a separate discriminative classifier (CNN in this case), and further compares its performance with the same CNN when it is trained on the original spectrogram inputs.

Impact of the latent space dimensionality

We first address the impact of latent space dimensionality on the ASVspoof 2017 corpus. To keep computation time manageable, we focus only on the C-VAE variant. The results, for both CQCC and spectrogram features, are summarised in Table 5.19. Shown results are on the ASVspoof 2017 v2.0 evaluation sets. We observe an overall decreasing trend in EER with increased latent space dimensionality, as expected. All the error rates are relatively high, which indicates general difficulty of our detection task. In the remainder of this study, we fix the latent space dimensionality to $d = 128$ as a suitable trade-off in EER and computational cost.

Table 5.20: Performance of GMM and VAE variants using CQCC inputs.

Model	ASVspoof 2017				ASVspoof 2019 PA			
	Dev		Eval		Dev		Eval	
	EER	t-DCF	EER	t-DCF	EER	t-DCF	EER	t-DCF
GMM	19.07	0.4365	22.6	0.6211	43.77	0.9973	45.48	0.9988
VAE	29.2	0.7532	32.37	0.8079	45.24	0.9855	45.53	0.9978
C-VAE	18.1	0.4635	**28.1**	**0.7020**	**34.06**	**0.8129**	36.66	0.9104
AC-VAE$_1$	21.8	0.4914	29.3	0.7365	34.73	0.8516	36.42	0.9036
AC-VAE$_2$	**17.78**	**0.4469**	29.73	0.7368	34.87	0.8430	**36.42**	**0.8963**

Comparing different VAE setups with GMMs

Our next experiment addresses the relative performance of different VAE variants and their relation to our GMM baseline. As GMMs cannot be used with high-dimensional spectrogram inputs, the results are shown only for the CQCC features. This experiment serves to answer the question on which VAE variants are the most promising, and whether VAEs have potential to be used as a back-end for spoofing detection. The results for both the ASVspoof 2017 and 2019 (PA) datasets are summarised in Table 5.20.

Baseline GMM. On the ASVspoof 2017 dataset, the GMM reports an EER of 19.07% and 22.6% on the development and evaluation sets, respectively. Note that our baseline is completely different from the CQCC-GMM results of [Delgado et al., 2018] for two reasons. First, we use a unified time representation of the first 100 frames obtained either by truncating or copying time frames, for reasons explained earlier. Second, we remove the leading and trailing non-speech/silence from every utterance, to mitigate a dataset-related bias described in Subsection 4.6.3: the goal of our modified setup is to ensure that our models focus on actual factors rather than database artefacts.

On the ASVspoof 2019 PA dataset, the performance of the GMM baseline[20] is nearly random as indicated by both metrics. The difficulty of the task and our modified setup to suppress database artefacts both contribute to high error rates. The results are consistent with our earlier findings in Subsection 5.2.5. The two separate GMMs may have learnt similar data distributions. Note that the similarly-trained naive VAE displays similar near-random performance.

VAE variants. Let us first focus on the ASVspoof 2017 results. Our first, naive VAE approach falls systematically behind our baseline GMM. Even if

[20]For sanity check, we trained a GMM without removing silence (and using all frames per utterance) and obtained a performance similar to the official GMM baseline of the ASVspoof 2019 challenge. On the development set, our GMM now shows an EER of 10.06% and t-DCF of 0.1971 which is slightly worse than official baseline (EER = 9.87 and t-DCF =0.1953).

Table 5.21: Comparing VAE and C-VAE performance on the ASVspoof 2017 dataset using the log power spectrograms as input features.

	Dev		Eval	
Model	EER	t-DCF	EER	t-DCF
VAE	32.12	0.8037	36.92	0.9426
C-VAE	22.81	0.5219	29.52	0.7302

both the bonafide and spoof VAE models display decent reconstructions, they lack the ability to retain discriminative information in the latent space when trained in isolation from each other. Further, remembering that VAE training optimises only a lower bound of the log-likelihood function, it might happen that the detection score formed as a difference of these 'inexact' log-likelihoods either under or overshoots the true log-likelihood ratio — but there is no way of knowing which way it is.

The C-VAE model, however, shows encouraging results compared with all the other VAE variants considered. This suggests that conditioning both the encoder and decoder with class labels during VAE training is helpful. Supposedly a shared, conditional C-VAE model yields 'more compatible' bonafide and spoof scores when we form the detection score. The C-VAE model shows comparable detection performance to the GMM baseline on the development set, though it performs poorly on the evaluation set.

The VAE variants with an auxiliary classifier outperform the naive VAE but are behind C-VAE: both AC-VAE$_1$ and AC-VAE$_2$ shows slightly degraded performance over C-VAE on the evaluation set. While AC-VAE$_1$ and AC-VAE$_2$ show comparable performance on the evaluation set, on the development set AC-VAE$_2$ outperforms all other VAE variants in both the metrics. This suggests overfitting on the development set: adding an auxiliary classifier increases the model complexity as the number of free parameters to be learned increases substantially. Apart from having to learn optimal model parameters from a small training dataset, another challenge is to find an optimal value for the control parameters α and β in Equation 5.2.

On the ASVspoof 2019[21] dataset our C-VAE model now outperforms the naive VAE and the GMM baseline. By conditioning the encoder and decoder networks with class labels, we observe an absolute improvement of about 10% over the naive VAE on both the development and the evaluation sets. Unlike in the ASVspoof 2017 dataset, the auxiliary classifier VAE now offers some improvement on the evaluation set. This might be due to the much larger

[21] We would like to stress that we do not use the original training and development protocols for model training and validation. Instead, we use our custom protocol described in Section 5.2.3 that helped to improve generalisation during the ASVspoof 2019 challenge. However, during testing, we report test results on the standard development and evaluation protocols.

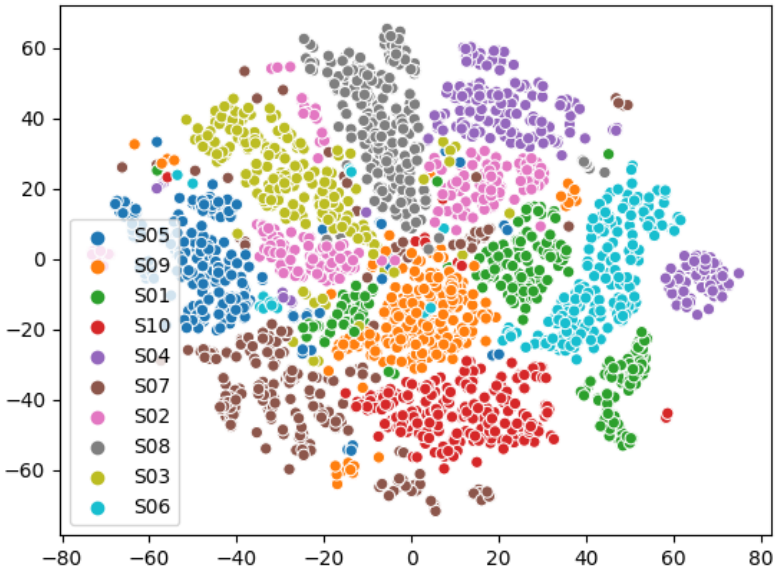


Figure 5.8: Latent space visualisation for 10 phrases S01-S10 (see Table 3.1) of the ASVspoof 2017 training set by C-VAE with bonafide-class conditioning.

number of training examples available in the ASVspoof 2019 dataset (54000 utterances) in comparison to the ASVspoof 2017 training set (3014 utterances).

It should be further noted that while training models on the ASVspoof 2019 dataset, we used the hyper-parameters (learning rate, mini-batch size, control parameters including the network architecture) that were optimised on the ASVspoof 2017 dataset. This was done to study how well the architecture and hyper-parameters generalise from one replay dataset (ASVspoof 2017) to another one (ASVspoof 2019).

The results in Table 5.20 with the CQCC features indicate that the C-VAE is the most promising variant for further experiments. While adding the auxiliary classifier improved performance in a few cases, the improvements are modest relative to the added complexity. Therefore, in the remainder of this paper, we focus on the C-VAE unless otherwise stated. Also, we focus testing our ideas on the ASVspoof 2017 replay dataset for computational reasons. Next, to confirm the observed performance improvement of the C-VAE over the naive VAE, we further train both models using raw log power-spectrogram features. The results in Table 5.21 confirm the anticipated result in terms of both metrics.

Conditioning VAEs beyond class labels

The results so far confirm that the C-VAE outperforms the naive VAE by a wide margin. We now focus on multi-class conditioning using C-VAEs. To this end, our possible conditioning variables could include speaker and sentence identifiers. However, speakers are different across the training and test sets in both ASVspoof 2017 and ASVspoof 2019, preventing the use of speaker conditioning. Further, the phrase identities of the ASVspoof 2019 PA dataset are not publicly available. For these reasons we restrict our focus on the 10 common passphrases (S01 through S10) in the ASVspoof 2017 dataset shared across training, development and evaluation data. The contents of these phrases are provided in Table 3.1. The numbers of bonafide and spoof utterances for these passphrases in the training and development sets are equally balanced. We therefore use a 20-dimensional one-hot vector to represent multi-class conditioning. The first 10 labels correspond to bonafide sentences S01 through S10 and the remaining 10 to spoofed utterances. Everything else about training and scoring the C-VAE remains the same as before, except for the use of the 20-dimensional (rather than 2-dimensional) one-hot vector.

We first visualise how the latent space is distributed across the 10 different phrases of the ASVspoof 2017 training set. Fig. 5.8 shows the 2D latent space visualisation for these utterances using the t-SNE [Maaten and Hinton, 2008] algorithm. The clear distinction between different phrases suggests that the latent space preserves the structure and identity of different sentences of the dataset. This suggests that choosing the sentence identity for conditioning the VAE might be beneficial towards improving performance; such a model is expected to learn *phrase-specific* bonafide-vs-spoof discriminatory cues.

Table 5.22 summarises the results. Shown results are on the ASVspoof 2017 v2.0 dataset using CQCC and spectrogram inputs. The C-VAE trained on spectrogram features with multi-class conditioning shows a substantial improvement over two-class conditioning. This suggests that the network now benefits from exploiting relevant information present across different passphrases, which may be difficult from binary class conditioning. For the CQCC features, however, we have the opposite finding: while the EER is slightly decreased on the evaluation set with multi-class conditioning, overall it shows degraded performance. One possible interpretation is that CQCCs are a compact feature representation optimised specifically for anti-spoofing. CQCCs may lack phrase-specific information relevant for anti-spoofing which is retained by the richer and higher-dimensional raw spectrogram. To sum up, the C-VAE trained on raw spectrograms with multi-class conditioning offers substantial improvement over two-class conditioning in comparison to CQCC input features.

Table 5.22: Comparing the performance of the C-VAE model trained using binary and multi-class conditioning.

	CQCC				Spectrogram			
	Dev		Eval		Dev		Eval	
Conditioning	EER	t-DCF	EER	t-DCF	EER	t-DCF	EER	t-DCF
Two-class	18.1	0.4635	28.1	0.7020	22.81	0.5219	29.52	0.7302
Multi-class	19.77	0.4961	27.88	0.7390	19.65	0.4324	25.48	0.6631

Qualitative results

A relevant question is whether or not the latent space features $\boldsymbol{z}$ have some clear meaning in terms of human or spoofed speech parameters, or any other relevant information that helps us derive some understanding about the underlying data. To this end, we analyse the latent space through 2D visualisations using the t-SNE algorithm. We aim to understand how the latent space is distributed across different speakers and between genders. We do this on the ASVspoof 2019 dataset, as the 2017 dataset only has male speakers. Fig. 5.9 shows t-SNE plots for 5 male and 5 female speakers on the ASVspoof 2019 PA training set chosen randomly.

The subfigures in the first row of Fig. 5.9 suggest that the latent space has learned quite well to capture speaker related information, but it does not seem to clearly capture gender specific information as evident from the overlapping clusters (top right figure). We further analyse bonafide and different attack conditions per gender, taking PA_0082 and PA_0079 — one male and female speaker randomly from the pool of 10 speakers we considered. Fig. 5.9, second row illustrates this. We use letters A-I to indicate the bonafide class and 9 different attack conditions whose original labels are as follows. A: bonafide, B: 'BB', C: 'BA', D: 'CC', E: 'AB', F: 'AC', G: 'AA', H: 'CA', I: 'CB', J: 'BC'. These labels refer to the combination of different factors involved in a replay attack. For example, an attacker to ASV microphone distance, quality of recording device, room acoustics, quality of playback device etc. As we do not use them further in this work, their in-depth details are not provided here. Rather, we point the reader to see [Todisco et al., 2019.] for more details. From Fig. 5.9, we observe overlapping attacks within a cluster, and spread of these attacks across different clusters. The bonafide audio examples, denoted by letter A are heavily overlapped by various spoofed examples. This gives an intuition that the latent space is unable to preserve much discriminative information due to the nature of the task, and rather, it might be focusing on generic speech attributes such as acoustic content, speaker speaking style to name a few, to be able to generate a reasonable reconstruction — as depicted in Fig 5.7.

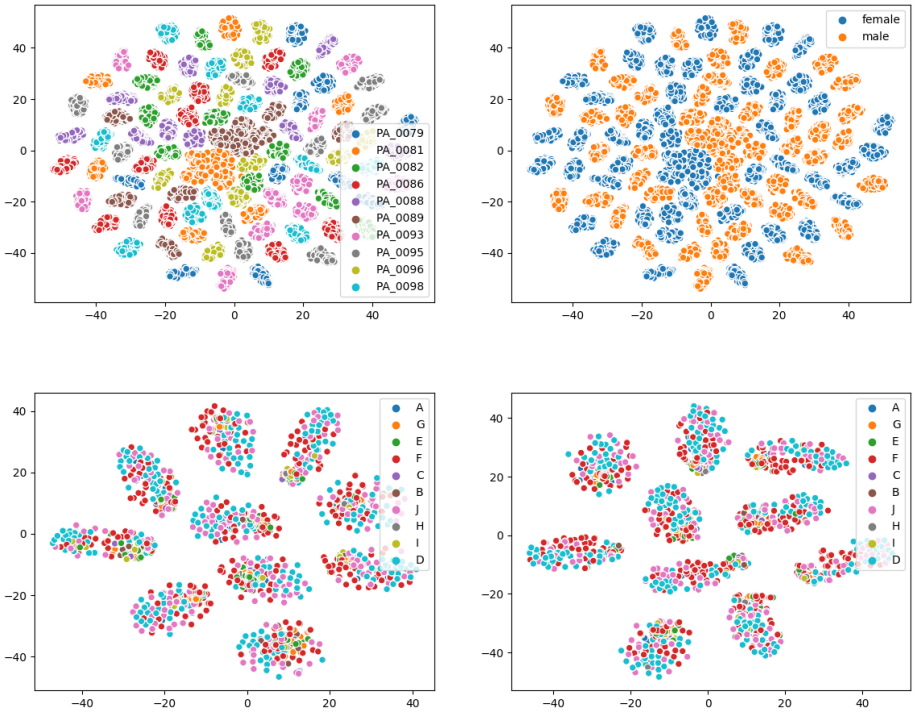


Figure 5.9: 2D visualisation of the latent space learned by the C-VAE. *Top left*: 10 different speaker identities. *Top right*: 5 male and 5 female clusters. *Bottom left*: distribution of bonafide and attack conditions for a male speaker PA_0082. *Bottom right*: same as in (c) but for a female speaker PA_0079.

Table 5.23: Effectiveness of VAE residual features in spoofing detection.

		Dev		Eval	
Features	Model	EER	t-DCF	EER	t-DCF
Spectrogram	C-VAE	22.81	0.5219	29.52	0.7302
VAE residual	CNN	13.16	0.3438	17.32	0.4293
Spectrogram	CNN	10.82	0.2877	16.03	0.4461

VAEs as feature extractors

Table 5.23 summarises the results. Results shown are on the ASVspoof 2017 v2.0 dataset. Numbers in the second row correspond to our proposed approach of using VAE residual features and training a separate classifier. We also include C-VAE results from our initial approach (C-VAE as a back-end) from the third row of Table 5.20 for comparison. For contrastive purposes we train another CNN classifier with the same architecture using the original spectrogram as its input. The results for this is shown in the third row of the table. Using

VAE residuals and training a separate classifier outperforms the back-end approach on both metrics and on both the development and evaluation sets. The residual approach, however, remains behind the CNN trained directly on the original spectrogram, on the development set. On the evaluation set, it achieves the lowest t-DCF and displays a comparable EER. The small performance gap (in relative terms) between the development and evaluation sets for the VAE residual approach suggests good generalisation.

Although the proposed VAE residual approach did not outperform the raw-spectrogram CNN, the results obtained are encouraging and show potential for further investigation. In fact, given the similar performance of the original and VAE residual spectrogram features, we interpret the results to mean that *most of* the relevant information for discriminating bonafide and replay utterances (on this data) lies in the residual or 'noise' part of the spectrogram. It is noteworthy that heuristic ideas inspired directly by simple visualisations such as Figs. 5.8 and 5.7 lead to boosted performance. Finally, recalling our initial motivations, the VAE leads to a generative model (unlike CNN) that allows data sampling and obtaining uncertainty of the latent space representation. These favorable properties of VAEs suggest further studies towards more versatile spoofing countermeasure solutions where the semantics, sanity and stability of the learned feature representation can be easily explored.

5.5.5 Discussion

Inspired by the successful use of GMMs, a classical generative model, as a backend in spoofing detection for ASV, this section performed an initial study on exploring the potential of VAEs, a deep generative model in audio spoofing detection. Subsection 5.5.2 described our proposed methods of using VAEs as classifiers, and as a feature extractor. In terms of backend classifier, different VAE variants: vanilla VAE, C-VAE and C-VAE with an auxiliary classifier were investigated. Subsection 5.5.4 described the evaluation of our methods.

Our first study using two separate VAEs suggests that it is difficult to capture discriminative information when the models are trained using only one-class data. Both the bonafide and spoof VAEs seem to focus on retaining information relevant for data reconstruction while giving less attention on class-discriminative information. As a result, the latent space in both bonafide and spoof VAEs appears to capture common prominent characteristics of bonafide and spoofed speech, making the detection task difficult. Nonetheless, our qualitative results indicate that both our bonafide and spoof VAEs yield reasonable reconstruction of the input data. Our second approach of training a single conditional VAE (C-VAE) by conditioning the encoder and decoder networks by

class-label vectors shows far more encouraging results. The performance of our C-VAE models on both the ASVspoof 2017 and ASVspoof 2019 datasets show remarkable improvement in comparison to the naive VAE approach. Our third approach of augmenting an auxiliary classifier with the C-VAE did not help much. We did not observe substantial improvement in detection performance on the ASVspoof 2017 dataset, though we observed some performance gain on the ASVspoof 2019 dataset, suggesting the importance of training set size for improved generalisation. Finally, our proposed approach of using C-VAE as a front-end (VAE residual features), demonstrated a substantial improvement over the VAE back-end use case.

Despite the different dataset sizes in the ASVspoof 2017 and ASVspoof 2019 datasets, we find that the model hyper-parameters tuned on the ASVspoof 2017 dataset worked quite well when applied on the 2019 dataset, showing consistency of our findings with C-VAE models. However, the optimisation of network architecture and model hyper-parameters has not been fully explored in the present study, leaving scope for further improvements. To sum up, based on both the observed detection performance and architecture complexity considerations, from the three VAE back-end variants considered (Fig. 5.5), the authors recommend potential future work to focus on conditional VAEs (C-VAEs). In fact, we obtained promising results by further conditioning C-VAEs using pass-phrase labels. This warrants future studies with other conditioning variables such as speaker identity, gender, and channel.

5.6 Summary

This chapter presented several novel methods for the design of countermeasures for replay spoofing detection while focussing on model robustness and avoiding potential biases in the datasets used for training and inference. Proposed methods were evaluated on two benchmark spoofing datasets: ASVspoof 2017 v2.0 and ASVspoof 2019 PA. The work reported in this chapter has been published (and is under review) in international peer-reviewed conferences and journals.

In Section 5.2, an ensemble model and a dataset partition was proposed for better generalisation. Its effectiveness was evaluated on a latest benchmark dataset (ASVspoof 2019 PA) demonstrating good performance on the test set. Artefacts in this dataset was discovered that provided substantial contributions in model predictions, and a method to mitigate this issue was proposed subsequently. Extending the findings from Section 4.6, the chapter proposed a method towards building reliable and trustworthy countermeasures on the ASVspoof 2017 v2.0 dataset. The method involved speech endpoint detection as a step before feature extraction to remove audio samples before and after

the actual speech utterance. The effectiveness of the proposed method was demonstrated by evaluating countermeasures (trained with and without endpoint detection) on manipulated test signals. Experimental results confirmed that countermeasures trained using the proposed method helped mitigate the impact substantially (Section 5.3).

The chapter then proposed a joint subband modeling framework (Section 5.4) designed to exploit information across different subbands independently. This method is evaluated on both the 2017 and 2019 PA benchmark datasets using the proposed data preprocessing steps removing the confounders in them. Furthermore, the proposed method is also evaluated in a cross-dataset test scenario using the ASVspoof 2019 real PA test set (described in Section 3.3.3). Although the experimental results confirmed the superiority of the proposed method over conventional method of training countermeasures on the fullband spectrum, the findings obtained on one dataset did not generalise across others indicating that the current datasets do not reflect real world replay conditions suggesting a need for careful dataset design.

Finally, the chapter proposed the use of VAE, a deep generative model, for replay spoofing detection in two different settings. First, as an alternative backend classifier, and as a feature extractor. However, unlike the traditional approaches used in the literature that use the latent space as a learned feature, this thesis used a different approach. The proposed feature, so called VAE residual, is obtained by taking the absolute difference between the original input and the model reconstruction. The proposed methods were evaluated on both the 2017 and 2019 PA datasets with the proposed preprocessing steps (Section 5.5). Experimental results demonstrated that training a single VAE with class conditioning (C-VAE) offered substantial improvement in comparison to training two separate VAEs for each class. Furthermore, the proposed frontend approach augmented with a CNN classifier demonstrated substantial improvement over the VAE backend use case.

Chapter 6

Conclusions and future work

This thesis aimed at the analysis and design of existing and novel methods for replay spoofing detection for secure voice biometrics. Towards achieving these objectives, several methods have been studied and proposed, and presented in two major chapters: Chapter 4 and 5. While Chapter 4 mostly focussed on the analysis part, Chapter 5 aimed at the design of novel spoofing countermeasures. Most of the work reported in the two chapters was published in peer-reviewed international journals and conferences as summarised in Section 1.5. This chapter provides a summary of the main contributions made from the thesis in Section 6.1, and discusses the potential future work for replay spoofing detection in Section 6.2.

6.1 Summary

6.1.1 Analysis of countermeasures

Chapter 4 aimed at serving as the basis towards understanding the replay spoofing attack problem. For this, a series of studies towards analysing replay spoofing countermeasures was presented by exploring existing methodologies and techniques from the literature.

The chapter first investigated the generalisability of signal processing frontends, which showed promising results in detecting converted and synthetic speech, for the task of replay spoofing detection using the ASVspoof 2017 (v1.0) benchmark dataset. Experimental results showed that obtaining the same level of generalisation is difficult due to the acoustically different task under consideration. Through the analysis of the best performing GMM countermeasure,

it was found that initial zero-valued silence, present in some of the bonafide recordings but missing from spoof class recordings, provided cues for class discrimination. Using this knowledge it was further demonstrated how easy it was to compromise countermeasure predictions. Though such data-intrinsic behavior may not appear in real-world scenarios, this work demonstrated the severe impact it can have on countermeasure performance on this dataset (Section 4.2).

Next, a detailed study on replicating a state-of-the-art deep model (LCNN) reported on the ASVspoof 2017 v1.0 dataset was performed, and it was shown that using only the published system details, reproducing this model was difficult. It was further shown that despite investigating various deep model architectures from the literature, achieving performance close to the one reported by the LCNN authors on the evaluation set was difficult. These models would often overfit on the validation data, resulting in a higher performance gap between the development and evaluation sets. Experimental results also showed that a 32 mini-batch size and ReLU non-linearity (which also outperformed the MFM activation used in the LCNN model) were optimal hyper-parameter choices among others to train deep models on this dataset (Section 4.3).

Afterwards, a performance analysis of several countermeasures under varied spoofing conditions was carried out using the updated version (2.0) of this dataset. Experimental results demonstrated that MFCCs were better at detecting low quality attacks while IMFCCs showed better results for high quality attacks. However, gaining an in-depth insight on what is causing such behaviour was difficult as the meta-data: acoustic environment - AE, recording devices - RD for the bonafide class was unavailable. This also suggests that "high-quality spoofing conditions" may actually be low quality since the bonafide files were of low quality and vice-versa. Therefore, on this dataset it was found that analysing factors (AE, RD, playback device and their interactions) influencing replay attacks in a controlled condition and providing a substantial conclusion whether reverberation noise or some device-specific (recording or playback) attributes provide a cue for replay spoofing detection was difficult (Section 4.4).

Next, a deep CNN countermeasure model was developed following guidelines from Section 4.3. It was found that batch normalisation was a key factor for model generalisation, which helped the CNN achieve performance closer to the state-of-the-art LCNN. The SLIME algorithm for generating local interpretable explanations was then applied on this model (trained on v2.0 of the dataset) to understand which part of the input highly influenced its prediction. Results showed that the CNN was giving high weights to the first few milliseconds of the audio signal to make class predictions. The significance of this analysis was further demonstrated by preprocessing the test signals which led to a predictable change in the system error rate, raising questions about the integrity

and trustworthiness of countermeasures trained on this dataset (Section 4.5).

Following this, the chapter then performed an in-depth analysis of audio recordings in the training and development sets (and bonafide recordings in the evaluation set) of the ASVspoof 2017 v2.0 dataset, identified artefacts (see Subsection 3.2.3) and investigated their impact on machine learning countermeasure decisions. Among different artefacts, burst click sound (BCS) was found to provide strong cues for the bonafide class. Interestingly, DTMF sounds showed no influence on model decisions (Subsection 4.6.3). Furthermore, zero-valued samples (silence) were still found to provide a potential cue for the bonafide class on the v2.0 of this dataset.

Our in-depth analysis of this dataset and experimental results from different interventions confirmed the presence of a "horse" in machine learning [Hernandez-Orallo, 2019, Sturm, 2014] for anti-spoofing applied to the ASVspoof 2017 dataset. Furthermore, none of the research results published in this dataset (more than 60 papers) have accounted for these artefacts, further indicating that the countermeasures showing impressive results may not be fully reliable and trustworthy [Sturm, 2016] as their decision process involved the contribution of these artefacts which are not related to the actual problem (Section 4.6).

6.1.2 Design of novel countermeasures

Replay spoofing attack detection, a binary classification problem, in general is a difficult task to solve. As explained in Subsection 2.9.1, confounders or artefacts in a dataset can affect a wide range of machine learning tasks including anti-spoofing systems (see Sections 4.2, 4.6 and Subsection 5.2.5). Such confounders are often overlooked in research studies. We consider two reasons for this. First, the figures of merit used in assessing performance (typically a scalar) do not account for confounders and their influence in learning algorithms. Second, they often offer gains in performance (see Subsection 2.9.1 for related background). Due to these reasons we often do not care towards the accountability of such ML models trained on data containing artefacts. But, impressive performance reported by such untrustworthy models can be costly as they may fail with high likelihood when used in practical real-world scenarios. Therefore, ensuring reliable performance estimates is important to truly assess the ability of proposed features/classifiers for a given machine learning task.

In this direction, Chapter 5 proposed novel methods towards design of countermeasures for replay spoofing attack detection while also focusing on model robustness and avoiding dataset biases. Firstly, in Section 5.2 an ensemble model and dataset partitions for better model generalisation and robustness were proposed and evaluated on a latest benchmark spoofing dataset (ASVspoof 2019).

The ensemble combined several deep and shallow models employing different features and training techniques for increased diversity leading to a powerful model. Although the proposed partition involved discarding a lot of spoofed data points in the training and validation sets, the experimental results demonstrated improved model robustness, which helped achieve good performance on the PA task and 3^{rd} ranking on the LA task of the ASVspoof 2019 challenge. Another key contribution included a demonstration of how countermeasures trained on the PA dataset can become somewhat of a "horse" [Sturm, 2014], where solving the actual problem is unintentionally avoided by exploiting "silence" as trivial cues, for which a simple pre-processing approach was proposed to mitigate the issue.

Then Section 5.3 proposed a method to mitigate the impact of artefacts identified in the ASVspoof 2017 v2.0 benchmark dataset (Section 4.6) and build robust CM models. For this, use of speech endpoint detection module before feature extraction was proposed to discard audio samples before and after the actual speech utterance. This ensures that both classes of audio now have a similar pattern, forcing learning algorithms to now focus exploiting factors of interest — for example channel characteristics, in solving the replay spoofing attack problem, thus producing reliable performance estimates. Manual speech endpoint annotations were developed and used during training and validation of model parameters to ensure the correctness of our proposed method. During testing, a robust voice activity detection algorithm was applied to obtain automatic endpoint annotations that showed satisfactory results when compared with manual annotations. Several new benchmark results (both frame-level and utterance-level countermeasures) are provided showcasing the true performance estimates when these confounders are taken into account. Finally, the robustness of countermeasures trained with and without endpoint detection was evaluated by manipulating test utterances with signal artefacts, and the proposed method was found to be more resilient confirming its robustness over countermeasures trained without endpoint detection on this dataset (Subsection 5.3.3).

In the next Section 5.4, a joint subband modelling framework was proposed and evaluated on the ASVspoof 2017 v2.0 and ASVspoof 2019 PA benchmark datasets. Signal preprocessing methods proposed in Section 5.2 and Section 5.3 were applied on both datasets to avoid biases during the training and testing (see Subsections 4.6.3 and 5.2.5). The proposed framework employed n sub-networks to learn subband specific features which were later combined and passed to a classifier, and the whole network weights were updated during training. Results on the ASVspoof 2017 dataset demonstrated that the first and the last 1 kHz frequency bands carried the most discriminative information, and the joint model trained on these two subbands showed the best performance outperform-

ing the baselines (trained on fullband spectra) by a large margin. However, these findings did not generalise on the ASVspoof 2019 PA dataset. Furthermore, models trained on the ASVspoof 2017 and 2019 PA datasets showed poor cross-dataset performance on the ASVspoof 2019 real PA test set, indicating that these datasets do not reflect real world replay conditions, suggesting a need for careful design and validation of replay spoofing datasets.

Finally, motivated from the widespread use of GMMs as a backend classifier, the chapter (Section 5.5) proposed VAEs as an alternative backend for replay attack detection, via three alternative models that differ in their class-conditioning. The first approach (vanilla VAE) was similar to that of traditional GMMs involving training of two separate VAEs — one for each class. The second approach trained a single conditional model (C-VAE) by injecting a one-hot class label vector to the encoder and decoder networks. The third approach involved integrating an auxiliary classifier to guide the learning of the latent space.

Quantitative results for the vanilla VAE suggested that it was difficult to capture discriminative information when the VAEs were trained using only one-class data. It was shown that both the bonafide and spoof models focused on retaining prominent characteristics of the speech signal relevant for data reconstruction while giving less attention on class-discriminative information. Nonetheless, the qualitative results indicated that both bonafide and spoof VAEs yielded a reasonable reconstruction of the input data. Quantitative results of the C-VAE model conditioning both the encoder and decoder networks by class-label vectors showed far more encouraging results. The performance of this model on both the ASVspoof 2017 and ASVspoof 2019 datasets showed remarkable improvement in comparison to the vanilla VAE models. The third approach of augmenting an auxiliary classifier with the C-VAE did not help much. Although this model offered some performance gains on the ASVspoof 2019 dataset, no substantial improvement was found on the ASVspoof 2017 dataset, suggesting the importance of training set size for improved generalisation.

Finally, this chapter proposed the use of VAE residuals — the absolute difference of the original input and the reconstruction — as a novel feature for replay spoofing detection and demonstrated substantial improvement over the VAE backend use case.

6.2 Future work

This section provides a summary of potential research directions that could be explored towards extending the work presented in this thesis, along with key issues and challenges in building trustworthy countermeasures for replay spoof-

ing attacks.

Analysis: on the reliability and trustworthiness of countermeasures

- Towards the *interpretability* of spoofing countermeasures. It would be interesting to apply the SLIME algorithm (Subsection 4.5.3) and other interpretability methods to investigate: how explanations vary across different phrases; how explanations vary across different types of replay conditions/configurations; and whether speaker information (for example, fundamental frequency, prosody) influences spoofing detection performance. As ASVspoof 2019 phrase IDs are not available, an automatic speech recognition system can be used to first decode the utterance ID and perform the study on phrase-based analysis. Another method from interpretable machine learning we aim to study towards understanding a deep countermeasure model is 'feature inversion' [Mishra et al., 2018] which helps derive a broader understanding of what information different hidden layers have captured about the bonafide and spoofed classes.

- Studies on fooling ML model decisions using adversarial inputs is an active research topic [Nguyen et al., 2015, Heaven, 2019, Szegedy et al., 2014]. Adversarial examples are carefully-crafted samples that are as real as the original input samples, imperceptible to humans, and are capable of easily manipulating model's trustworthiness [Yuan et al., 2017]. Therefore, it is important to ensure robustness of ML models against adversarial attacks. Spoofing countermeasures are primarily designed to enhance trustworthiness of voice biometrics to end-users. This also indicates that robustness of such systems in the face of adversarial attacks is of prime importance in achieving the goal of trustworthiness. Although there are a few recent works on adversarial attacks on anti-spoofing systems [Liu et al., 2019b], but these works are primarily focussed on countermeasures for LA attacks (speech synthesis and voice conversion). Investigating the robustness of replay spoofing countermeasures against adversarial attacks would be another research avenue we look forward to.

- To mitigate the issues identified on the ASVspoof 2019 PA dataset, we proposed a simple solution to remove the first block of zero-valued samples before and after spoofed utterances. However, near-silent segments and silences between words within the recording might remain providing potential cues for spoofing detection. While it is desirable to have ML models exploit them for discrimination but we also need to ensure that they do not leave loop-holes for manipulation by a simply copy-paste of silences as we demonstrated in this thesis (Subsections 4.2.3 and 5.2.5).

To this end, we also aim to investigate the tradeoff between robustness and accuracy by incorporating a voice activity detector to discard all non-speech/silence frames from the whole utterance. We aim to extend this study on the ASVspoof 2017 v2.0 dataset as well.

On the design of Countermeasures

- The ASVspoof 2017 dataset comprises ten different phrases (see Table 3.1). It would be interesting to analyse how different phrases and words relate to the detection performance of countermeasure models. To this end, deep CM models can be trained by conditioning the sentence ID to derive phrase-specific models. An alternative to this would be to incorporate *multi-task learning* [Caruana, 1997] with phrase ID detection being the secondary task and spoofing detection as the primary task to be learned simultaneously.

- To gauge the importance of each subband, Section 5.4 can be further extended by excluding one of the 1 kHz bands (out of 8 uniformly split bands, each of 1 kHz) at a time and train the model on the remaining 7 bands and test the performance, and repeating the process altering the band to be discarded. In this kind of ablation study, a high increase in EER (relative to the use of the fullband spectrum for training) will indicate the importance of that subband. Likewise, if the performance is almost the same, this would indicate the specific band is less useful for the task.

- As described in Section 5.5, the optimisation of network architecture and model hyper-parameters for the ASVspoof 2019 PA dataset has not been fully explored in the present study (as it used parameters optimised on the 2017 dataset), leaving scope for further improvements. We found promising results by conditioning VAEs using phrase IDs; this warrants future studies with other conditioning variables such as speaker identity, gender, and channel.

- Exploring the newly released ReMASC dataset (Subsection 3.4.1). Building new countermeasure models on this dataset and performing cross-dataset performance evaluation with the ASVspoof 2017 and 2019 PA datasets. Further analysing different attack conditions of the ReMASC evaluation set is one potential research avenue.

- As a future work we aim to extend our study using VAEs (Section 5.5) for the detection of synthetic and voice-converted speech on the ASVspoof 2015 and the ASVspoof 2019 LA datasets.

- Feature learning from raw data. The experimental results from this thesis suggested that data driven models or end-to-end models are a potential direction to pursue for replay spoofing detection due to the nature of the task. One potential future direction includes investigating end-to-end models to learn frame-level discriminative features from raw audio.

- In Section 5.2, we studied an ensemble model combining models at the score-level. It would be interesting to extend this work by combining information at the feature level. The idea is to combine learned features from deep models with hand-crafted features and train a DNN on these features. Deep models can be trained on different inputs (eg. spectrograms or raw audio) to learn diverse features which may potentially help during feature fusion. These can be further combined through late fusion (score-level fusion). This approach can be studied on both benchmark datasets.

ASVspoof challenges and dataset design. Having participated in the two editions of the ASVspoof challenges in 2017 and 2019 we identified *reliability, robustness and reproducibility* as key factors that need attention, which in-turn directly relate to dataset design issues that we identified on both the datasets. We make the following recommendations that can be incorporated in future ASVspoof and other related challenges to promote trustworthiness in anti-spoofing research and development.

First, *controlled dataset design and model validation* to ensure they are free from artefacts and confounders as we have identified in this thesis. One simple approach for this would be to analyse the frame-wise log-likelihood score distribution of few confidently classified audio examples by a GMM baseline (see Subsection 4.2.3). Alternatively, a baseline deep model could be trained using utterance-level feature representations and the method we adopted in Subsection 4.5.3 can be applied to validate if there are potential cues/artefacts in the dataset that models are exploiting to form predictions. These simple approaches can help ensure that the challenge datasets are clean and models trained on them would be free from dataset biases.

Second, *reproducibility* of the top performing systems for *transparency* should be promoted in future challenges. This could be incorporated by imposing compulsory code submission of top ranking systems, or a separate track for reproducibility could be introduced. Incorporating such constraints will of course introduce additional work, however, these recommendations will also help avoid additional work towards reproducing a model that is hard to do so by following missing details of the reported systems, thereby promoting transparency and trustworthiness in anti-spoofing research. For example, the popular LCNN model that demonstrated the best results in the ASVspoof 2017 challenge is a

mystery that no research group has been able to reproduce until to this date. Such mysteries can be avoided and research results can become more fruitful with the introduction of reproducibility guidelines.

Appendices

Appendix A

Deep model architectures

This appendix describes the architecture details of the deep models that were used in Chapters 4 and 5 of the thesis. More precisely, descriptions of the original LCNN architecture from Lavrentyeva et al. [2017] is provided for reference, a description of the adapted version of LCNN model referred as CNN_1 is provided, and the description of two additional deep architectures that was proposed in this thesis is also provided. In each of the architectures described below the abbreviations: Conv, Dense, MFM, and MP denotes convolutional, full connected, max feature map and max pooling, respectively.

Table A.1 summarises the LCNN model architecture that was used in [Lavrentyeva et al., 2017]. This architecture has been used in Sections 4.2, 4.3, and 4.4 with different input dimensions (defined in time (T) and frequency (F)). The original LCNN uses $T = 400$ and $F = 864$.

Table A.2 summarises the CNN_1 model architecture which is adapted from LCNN (Table A.1), and is used in Sections 4.5, 4.6 and in all the sections of Chapter 5. CNN_1 applies a batch normalisation and ReLU nonlinearity after every Conv and Dense layers unlike LCNN that uses MFM nonlinearity without batch normalisation. A dropout layer with 50% drop ratio is applied before the Dense_1 layer and dense_2 layer applies sigmoid nonlinearity.

Table A.3 summarises the CNN_2 model architecture that is used in Sections 4.6 and 5.3. And, Table A.4 summarises the frame-based DNN model architecture that is used in Section 5.3.

Table A.1: The generalised LCNN architecture that operates on the input of shape $F \times T \times 1$. The original LCNN [Lavrentyeva et al., 2017] implementation uses $T = 400$ and $F = 864$.

Layer	**Input shape**	**Filter size**	**Stride size**	**# Filters/ neurons**	**Output shape**
Conv1	$F \times T \times 1$	5×5	1×1	32	$F \times T \times 32$
MFM1	$F \times T \times 32$	-	-	-	$F \times T \times 16$
MP1	$F \times T \times 16$	2×2	2×2	-	$F/2 \times T/2 \times 16$
Conv2a	$F/2 \times T/2 \times 16$	1×1	1×1	32	$F/2 \times T/2 \times 32$
MFM2a	$F/2 \times T/2 \times 32$	-	-	-	$F/2 \times T/2 \times 16$
Conv2b	$F/2 \times T/2 \times 16$	3×3	1×1	48	$F/2 \times T/2 \times 48$
MFM2b	$F/2 \times T/2 \times 48$	-	-	-	$F/2 \times T/2 \times 24$
MP2	$F/2 \times T/2 \times 24$	2×2	2×2	-	$F/4 \times T/4 \times 24$
Conv3a	$F/4 \times T/4 \times 24$	1×1	1×1	48	$F/4 \times T/4 \times 48$
MFM3a	$F/4 \times T/4 \times 48$	-	-	-	$F/4 \times T/4 \times 24$
Conv3b	$F/4 \times T/4 \times 24$	3×3	1×1	64	$F/4 \times T/4 \times 64$
MFM3b	$F/4 \times T/4 \times 64$	-	-	-	$F/4 \times T/4 \times 32$
MP3	$F/4 \times T/4 \times 32$	2×2	2×2	-	$F/8 \times T/8 \times 32$
Conv4a	$F/8 \times T/8 \times 32$	1×1	1×1	64	$F/8 \times T/8 \times 64$
MFM4a	$F/8 \times T/8 \times 64$	-	-	-	$F/8 \times T/8 \times 32$
Conv4b	$F/8 \times T/8 \times 32$	3×3	1×1	32	$F/8 \times T/8 \times 32$
MFM4b	$F/8 \times T/8 \times 32$	-	-	-	$F/8 \times T/8 \times 16$
MP4	$F/8 \times T/8 \times 16$	2×2	2×2	-	$F/16 \times T/16 \times 16$
Conv5a	$F/16 \times T/16 \times 16$	1×1	1×1	32	$F/16 \times T/16 \times 32$
MFM5a	$F/16 \times T/16 \times 32$	-	-	-	$F/16 \times T/16 \times 16$
Conv5b	$F/16 \times T/16 \times 16$	3×3	1×1	32	$F/16 \times T/16 \times 32$
MFM5b	$F/16 \times T/16 \times 32$	-	-	-	$F/16 \times T/16 \times 16$
MP5	$F/16 \times T/16 \times 16$	2×2	2×2	-	$F/32 \times T/32 \times 16$
FC6	$F/32 \times T/32 \times 16$	-	-	64	32×2
MFM6	32×2	-	-	32	32
FC7	32	-	-	2	2

Table A.2: CNN_1 model architecture which is adapted from Table A.1. This model has $188,875$ free parameters. * indicates batch normalisation and ReLU non-linearity. A dropout layer with 50% drop ratio is applied before the $Dense_1$ layer. The $Dense_2$ layer uses sigmoid non-linearity.

Layer	**Input shape**	**Filter size**	**Stride size**	**# Filters/ neurons**	**Output shape**
$Conv_1$	$865 \times 400 \times 1$	5×5	1×1	16	$865 \times 400 \times 16$
MP_1	$865 \times 400 \times 16$	2×2	2×2	-	$432 \times 200 \times 16$
$Conv_2^*$	$432 \times 200 \times 16$	3×3	1×1	16	$432 \times 200 \times 16$
$Conv_3^*$	$432 \times 200 \times 16$	3×3	1×1	24	$432 \times 200 \times 24$
MP_2	$432 \times 200 \times 24$	2×2	2×2	-	$216 \times 100 \times 24$
$Conv_4^*$	$216 \times 100 \times 24$	3×3	1×1	32	$216 \times 100 \times 32$
$Conv_5^*$	$216 \times 100 \times 32$	3×3	1×1	32	$216 \times 100 \times 32$
MP_3	$216 \times 100 \times 32$	2×2	2×2	-	$108 \times 50 \times 32$
$Conv_6^*$	$108 \times 50 \times 32$	3×3	1×1	32	$108 \times 50 \times 32$
$Conv_7^*$	$108 \times 50 \times 32$	3×3	1×1	16	$108 \times 50 \times 16$
MP_4	$108 \times 50 \times 16$	2×2	2×2	-	$54 \times 25 \times 16$
$Conv_8^*$	$54 \times 25 \times 16$	3×3	1×1	16	$54 \times 25 \times 16$
$Conv_9^*$	$54 \times 25 \times 16$	3×3	1×1	16	$54 \times 25 \times 16$
MP_5	$54 \times 25 \times 16$	2×2	2×2	-	$27 \times 12 \times 16$
Flatten	$27 \times 12 \times 16$	-	-	-	5184
$Dense_1^*$	5184	-	-	32	32
$Dense_2$	32	-	-	1	1

Table A.3: CNN_2 model architecture. This model has only 36,174 free parameters. * has the same meaning as in Table A.2. A dropout layer with 50% drop ratio is applied before the $Dense_1$ layer, and $Dense_2$ applies sigmoid non-linearity.

Layer	**Input shape**	**Filter size**	**Stride size**	**# Filters/ neurons**	**Output shape**
$Conv_1^*$	$257 \times 300 \times 1$	3×3	1×1	20	$257 \times 300 \times 20$
MP_1	$257 \times 300 \times 20$	3×3	1×1	-	$85 \times 100 \times 20$
$Conv_2^*$	$85 \times 100 \times 20$	3×3	1×1	15	$85 \times 100 \times 15$
MP_2	$85 \times 100 \times 15$	3×3	1×1	-	$28 \times 33 \times 15$
$Conv_3^*$	$28 \times 33 \times 15$	3×3	1×1	10	$28 \times 33 \times 10$
MP_3	$28 \times 33 \times 10$	3×3	1×1	-	$9 \times 11 \times 10$
Flatten	$9 \times 11 \times 10$	-	-	-	990
$Dense_1^*$	990	-	-	32	32
$Dense_2$	32	-	-	1	1

Table A.4: Frame-based DNN architecture. ReLU non linearity is applied in all dense layers except Dense_4, the output layer that uses sigmoid non linearity. A batch normalisation is applied to every layer before applying non linearity. Furthermore, a dropout layer is applied before every dense layer with 30% to the inputs of Dense_1 and 50% to the other three dense layer inputs.

Layer	**Input shape**	**# Neurons**	**Output shape**
Flatten	$1 \times 257 \times 1$	-	257
Dense_1	257	256	256
Dense_2	256	128	128
Dense_3	128	32	32
Dense_4	32	1	1

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

AI-HLEG. Ethics Guidelines for Trustworthy AI, 2020. URL `https://ec.europa.eu/futurium/en/ai-alliance-consultation/guidelines`.

K.N.R.K. Raju Alluri and Anil Kumar Vuppala. IIIT-H Spoofing Countermeasures for Automatic Speaker Verification Spoofing and Countermeasures Challenge 2019. In *Proc. Interspeech 2019*, pages 1043–1047, 2019. doi: 10.21437/Interspeech.2019-1623. URL `http://dx.doi.org/10.21437/Interspeech.2019-1623`.

K.N.R.K. Raju Alluri, Sivanand Achanta, Sudarsana Reddy Kadiri, Suryakanth V. Gangashetty, and Anil Kumar Vuppala. SFF Anti-Spoofer: IIIT-H Submission for Automatic Speaker Verification Spoofing and Countermeasures Challenge 2017. In *Proc. Interspeech 2017*, pages 107–111, 2017. doi: 10.21437/Interspeech.2017-676. URL `http://dx.doi.org/10.21437/Interspeech.2017-676`.

Moustafa Alzantot, Ziqi Wang, and Mani B. Srivastava. Deep Residual Neural Networks for Audio Spoofing Detection. In *Proc. Interspeech 2019*, pages 1078–1082, 2019. doi: 10.21437/Interspeech.2019-3174. URL `http://dx.doi.org/10.21437/Interspeech.2019-3174`.

ASVspoof 2019 evaluation_plan. The Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan. URL http://www.asvspoof.org/asvspoof2019/asvspoof2019_evaluation_plan.pdf.

Anderson R. Avila, Jahangir Alam, Douglas O'Shaughnessy, and Tiago H. Falk. Blind Channel Response Estimation for Replay Attack Detection. In *Proc. Interspeech 2019*, pages 2893–2897, 2019. doi: 10.21437/Interspeech.2019-2956. URL http://dx.doi.org/10.21437/Interspeech.2019-2956.

Bekir Bakar and Cemal Hanilçi. An Experimental Study on Audio Replay Attack Detection Using Deep Neural Networks. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 132–138, 2018.

Christian Bartz, Tom Herold, Haojin Yang, and Christoph Meinel. Language Identification Using Deep Convolutional Recurrent Neural Networks. *arXiv preprint arXiv:1708.04811*, 2017.

Laurent Besacier and Jean-François Bonastre. Subband architecture for automatic speaker recognition. *Signal Processing*, 80(7):1245 – 1259, 2000. ISSN 0165-1684. doi: https://doi.org/10.1016/S0165-1684(00)00033-5. URL http://www.sciencedirect.com/science/article/pii/S0165168400000335.

Radosław Białobrzeski, Michał Kośmider, Mateusz Matuszewski, Marcin Plata, and Alexander Rakowski. Robust Bayesian and Light Neural Networks for Voice Spoofing Detection. In *Proc. Interspeech 2019*, pages 1028–1032, 2019. doi: 10.21437/Interspeech.2019-2676. URL http://dx.doi.org/10.21437/Interspeech.2019-2676.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Merlijn Blaauw and Jordi Bonada. Modeling and Transforming Speech using Variational Autoencoders. In *Proc. Interspeech 2016*, pages 1770–1774, Sept 2016. doi: 10.21437/Interspeech.2016-1183.

Jean-francois Bonastre, Driss Matrouf, and Corinne Fredouille. Transfer Function-Based Voice Transformation for Speaker Recognition. In *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop*, pages 1–6, 2006.

Niko Brümmer and Edward de Villiers. The bosaris toolkit: Theory, algorithms and code for surviving the new dcf. *arXiv preprint arXiv:1304.2865*, 2013.

David K. Burton. Text-dependent speaker verification using vector quantization source coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(2):133–143, 1987.

Weicheng Cai, Danwei Cai, Wenbo Liu, Gang Li, and Ming Li. Countermeasures for Automatic Speaker Verification Replay Spoofing Attack : On Data Augmentation, Feature Representation, Classification and Fusion. In *Proc. Interspeech 2017*, pages 17–21, 2017. doi: 10.21437/Interspeech.2017-906. URL `http://dx.doi.org/10.21437/Interspeech.2017-906`.

Weicheng Cai, Haiwei Wu, Danwei Cai, and Ming Li. The DKU Replay Detection System for the ASVspoof 2019 Challenge: On Data Augmentation, Feature Representation, Classification, and Fusion. In *Proc. Interspeech 2019*, pages 1023–1027, 2019. doi: 10.21437/Interspeech.2019-1230. URL `http://dx.doi.org/10.21437/Interspeech.2019-1230`.

William M. Campbell, Joseph P. Campbell, Douglas A. Reynolds, Elliot Singer, and Pedro A. Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech & Language*, 20(2):210 – 229, 2006. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2005.06.003. URL `http://www.sciencedirect.com/science/article/pii/S0885230805000318`.

William M. Campbell, Joseph P. Campbell, Terry P. Gleason, Douglas A. Reynolds, and Wade Shen. Speaker Verification Using Support Vector Machines and High-Level Features. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2085–2094, 2007.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41 – 75, July 1997. doi: https://doi.org/10.1023/A:1007379606734.

Sandipan Chakroborty, Anindya Roy, and Goutam Saha. Improved closed set text-independent speaker identification by combining MFCC with evidence from flipped filter banks. *IJSP*, 4(2):114–122, 2007.

Teck Kai Chan, Cheng Siong Chin, and Ye Li. Non-Negative Matrix Factorization-Convolutional Neural Network (NMF-CNN) For Sound Event Detection. *arXiv preprint arXiv:2001.07874*, 2020.

Su-Yu Chang, Kai-Cheng Wu, and Chia-Ping Chen. Transfer-Representation Learning for Detecting Spoofing Attacks with Converted and Synthesized Speech in Automatic Speaker Verification System. In *Proc. Interspeech 2019*, pages 1063–1067, 2019. doi: 10.21437/Interspeech.2019-2014. URL `http://dx.doi.org/10.21437/Interspeech.2019-2014`.

Christoforos C. Charalambous and Anil A. Bharath. A data augmentation methodology for training machine/deep learning gait recognition algorithms. *arXiv preprint arXiv:1610.07570*, 2016.

Jonathan Chen and Steven Asch. Machine Learning and Prediction in Medicine — Beyond the Peak of Inflated Expectations. *New England Journal of Medicine*, 376:2507–2509, 06 2017. doi: 10.1056/NEJMp1702071.

Zhuxin Chen, Zhifeng Xie, Weibin Zhang, and Xiangmin Xu. ResNet and Model Fusion for Automatic Spoofing Detection. In *Proc. Interspeech 2017*, pages 102–106, 2017. doi: 10.21437/Interspeech.2017-1085. URL `http://dx.doi.org/10.21437/Interspeech.2017-1085`.

Bhusan Chettri and Bob L. Sturm. A Deeper Look at Gaussian Mixture Model Based Anti-Spoofing Systems. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5159–5163, 2018.

Bhusan Chettri, Saumitra Mishra, Bob L. Sturm, and Emmanouil Benetos. Analysing the Predictions of a CNN-based Replay Spoofing Detection System. In *IEEE International Workshop on Spoken Language Technology (SLT)*, pages 92–97, September 2018a.

Bhusan Chettri, Saumitra Mishra, Bob L. Sturm, and Emmanouil Benetos. A Study On Convolutional Neural Network Based End-To-End Replay Anti-Spoofing. *arXiv preprint arXiv:1805.09164*, 2018b.

Bhusan Chettri, Bob L. Sturm, and Emmanouil Benetos. Analysing Replay Spoofing Countermeasure Performance under Different Conditions. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018c.

Bhusan Chettri, Daniel Stoller, Veronica Morfi, Marco A. Martínez Ramírez, Emmanouil Benetos, and Bob L. Sturm. Ensemble Models for Spoofing Detection in Automatic Speaker Verification. In *Proc. Interspeech 2019*, pages 1018–1022, September 2019.

Bhusan Chettri, Tomi Kinnunen, and Emmanouil Benetos. Deep generative variational autoencoding for replay spoof detection in automatic speaker verification. *Computer Speech & Language*, 63:101092, 2020. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2020.101092. URL `http://www.sciencedirect.com/science/article/pii/S0885230820300255`.

François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2001. ISBN 9780471062592. doi: 10.1002/0471200611. URL https://doi.org/10.1002/0471200611.

Sara Dahmani, Vincent Colotte, Valérian Girard, and Slim Ouni. Conditional Variational Auto-Encoder for Text-Driven Expressive AudioVisual Speech Synthesis. In *Proc. Interspeech 2019*, pages 2598–2602, 2019. doi: 10.21437/Interspeech.2019-2848. URL http://dx.doi.org/10.21437/Interspeech.2019-2848.

Rohan Kumar Das, Jichen Yang, and Haizhou Li. Long Range Acoustic Features for Spoofed Speech Detection. In *Proc. Interspeech 2019*, pages 1058–1062, 2019. doi: 10.21437/Interspeech.2019-1887. URL http://dx.doi.org/10.21437/Interspeech.2019-1887.

Steven B. Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, Aug 1980.

Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, May 2011.

Hector Delgado, Massimiliano Todisco, Mohammad. Sahidullah, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, and Junichi Yamagishi. ASVspoof 2017 Version 2.0: meta-data analysis and baseline enhancements. In *Proc. Speaker Odyssey*, 2018.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977. URL http://web.mit.edu/6.435/www/Dempster77.pdf.

Srinivas Desai, E. Veera Raghavendra, B. Yegnanarayana, Alan W. Black, and Kishore Prahallad. Voice conversion using Artificial Neural Networks. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3893–3896, 2009.

Bradley W. Dickinson and Kenneth Steiglitz. Eigenvectors and functions of the discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 30(1):25–31, 1982.

Heinrich Dinkel, Nanxin Chen, Yanmin Qian, and Kai Yu. End-to-end spoofing detection with raw waveform CLDNNS. In *Proc. ICASSP*, pages 4860–4864, March 2017.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

Serife Kucur Ergünay, Elie Khoury, Alexandros Lazaridis, and Sébastien Marcel. On the vulnerability of speaker verification to realistic voice spoofing. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–6, Sep. 2015. doi: 10.1109/BTAS.2015.7358783.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing Higher-Layer Features of a Deep Network. *Technical Report, University of Montreal*, 01 2009.

Philippe Esling, Axel Chemla–Romeu-Santos, and Adrien Bitton. Generative timbre spaces with variational audio synthesis. In *Proc. of the 21st International Conference on Digital Audio Effects*, 2018.

Fuming Fang, Junichi Yamagishi, Isao Echizen, and Jaime Lorenzo-Trueba. High-quality nonparallel voice conversion based on cycle-consistent adversarial network. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5279–5283, 2018a.

Fuming Fang, Junichi Yamagishi, Isao Echizen, Mohammad Sahidullah, and Tomi Kinnunen. Transforming acoustic characteristics to deceive playback spoofing countermeasures of speaker verification systems. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–9, Dec 2018b. doi: 10.1109/WIFS.2018.8630764.

Siyuan Feng and Tan Lee. Improving Unsupervised Subword Modeling via Disentangled Speech Representation Learning and Transformation. In *Proc. Interspeech 2019*, pages 281–285, 2019. doi: 10.21437/Interspeech.2019-1338. URL `http://dx.doi.org/10.21437/Interspeech.2019-1338`.

Roberto Font, Juan M. Espín, and María José Cano. Experimental Analysis of Features for Replay Attack Detection — Results on the ASVspoof 2017 Challenge. In *Proc. Interspeech 2017*, pages 7–11, 2017. doi: 10.21437/Interspeech.2017-450. URL `http://dx.doi.org/10.21437/Interspeech.2017-450`.

Daniel Garcia-Romero, David Snyder, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur. x-Vector DNN Refinement with Full-Length Recordings for Speaker Recognition. In *Proc. Interspeech 2019*, pages 1493–1496,

2019. doi: 10.21437/Interspeech.2019-2205. URL http://dx.doi.org/10.21437/Interspeech.2019-2205.

Sachin Garg, Shruti Bhilare, and Vivek Kanhanga. Subband Analysis for Performance Improvement of Replay Attack Detection in Speaker Verification Systems. In *2019 IEEE 5th International Conference on Identity, Security, and Behavior Analysis (ISBA)*, pages 1–7, Jan 2019.

Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep Voice 2: Multi-Speaker Neural Text-to-Speech. In *Advances in Neural Information Processing Systems 30*, pages 2962–2970, 2017.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 249–256, 2010.

Alejandro Gomez-Alanis, Antonio M. Peinado, Jose A. Gonzalez, and Angel M. Gomez. A Gated Recurrent Convolutional Neural Network for Robust Spoofing Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):1985–1999, 2019a.

Alejandro Gomez-Alanis, Antonio M. Peinado, Jose A. Gonzalez, and Angel M. Gomez. A Light Convolutional GRU-RNN Deep Feature Extractor for ASV Spoofing Detection. In *Proc. Interspeech 2019*, pages 1068–1072, 2019b. doi: 10.21437/Interspeech.2019-2212. URL http://dx.doi.org/10.21437/Interspeech.2019-2212.

Yuan Gong, Jian Yang, Jacob Huber, Mitchell MacKnight, and Christian Poellabauer. ReMASC: Realistic Replay Attack Corpus for Voice Controlled Systems. In *Proc. Interspeech 2019*, pages 2355–2359, 2019. doi: 10.21437/Interspeech.2019-1541. URL http://dx.doi.org/10.21437/Interspeech.2019-1541.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing sys- tems*, pages 2672–2680, 2014.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Thomas Grill and Jan Schlüter. Two convolutional neural networks for bird detection in audio signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1764–1768, Aug 2017.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. PixelVAE: A Latent Variable Model for Natural Images. *arXiv preprint arXiv:1611.05013*, 2016.

Tharshini Gunendradasan, Buddhi Wickramasinghe, Ngoc Phu Le, Eliathamby Ambikairajah, and Julien Epps. Detection of Replay-Spoofing Attacks Using Frequency Modulation Features. In *Proc. Interspeech 2018*, pages 636–640, 2018. doi: 10.21437/Interspeech.2018-1473. URL `http://dx.doi.org/10.21437/Interspeech.2018-1473`.

Tharshini Gunendradasan, Eliathamby Ambikairajah, Julien Epps, and Haizhou Li. An Adaptive-Q Cochlear Model for Replay Spoofing Detection. In *Proc. Interspeech 2019*, pages 2918–2922, 2019. doi: 10.21437/Interspeech.2019-2361. URL `http://dx.doi.org/10.21437/Interspeech.2019-2361`.

Vishwa Gupta, Patrick Kenny, Pierre Ouellet, and Themos Stafylakis. I-vector-based speaker adaptation of deep neural networks for French broadcast audio transcription. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6334–6338, 2014.

Rosa González Hautamäki, Tomi Kinnunen, Ville Hautamäki, and Anne-Maria Laukkanen. Automatic versus human speaker verification: The case of voice mimicry. *Speech Communication*, 72:13 – 31, 2015. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2015.05.002. URL `http://www.sciencedirect.com/science/article/pii/S0167639315000503`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*, 2015a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv preprint arXiv:1502.01852*, 2015b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016.

Douglas Heaven. Why deep-learning AIs are so easy to fool. *Nature*, 574:163–166, 10 2019. doi: 10.1038/d41586-019-03013-5.

Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119, 2016.

Jose Hernandez-Orallo. Gazing into Clever Hans machines. *Nature Machine Intelligence*, page 1, 03 2019. doi: 10.1038/s42256-019-0032-5.

Wei Ning Hsu, Yu Zhang, and James Glass. Learning Latent Representations for Speech Generation and Transformation. In *Proc. Interspeech*, pages 1273–1277, 2017a.

Wei Ning Hsu, Yu Zhang, and James Glass. Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data. In *Advances in Neural Information Processing Systems*, 2017b.

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018.

Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis. In *Proc. of the* 32^{nd} *International Conference on Neural Information Processing Systems*, NIPS'18, pages 52–63, USA, 2018. Curran Associates Inc.

Zhen Huang, Tim Ng, Leo Liu, Henry Mason, Xiaodan Zhuang, and Daben Liu. SNDCNN: Self-normalizing deep CNNs with scaled exponential linear units for speech recognition. *arXiv preprint arXiv:1910.01992*, 2019.

Andrew J. Hunt and Alan W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376, 1996.

Sergey Ioffe. Probabilistic Linear Discriminant Analysis. In *Computer Vision – ECCV 2006*, pages 531–542, Berlin, Heidelberg, 2006.

ISO/IEC. Information technology — Biometric presentation attack detection — Part 1: Framework. `https://www.iso.org/obp/ui/#iso:std:iso-iec:30107:-1:ed-1:v1:en`, 2016.

Sarfaraz Jelil, Rohan Kumar Das, S.R. Mahadeva Prasanna, and Rohit Sinha. Spoof Detection Using Source, Instantaneous Frequency and Cepstral Features. In *Proc. Interspeech 2017*, pages 22–26, 2017. doi: 10.21437/Interspeech.2017-930. URL `http://dx.doi.org/10.21437/Interspeech.2017-930`.

Sarfaraz Jelil, Sishir Kalita, S R Mahadeva Prasanna, and Rohit Sinha. Exploration of Compressed ILPR Features for Replay Attack Detection. In *Proc.*

Interspeech 2018, pages 631–635, 2018. doi: 10.21437/Interspeech.2018-1297. URL http://dx.doi.org/10.21437/Interspeech.2018-1297.

Zhe Ji, Zhi-Yi Li, Peng Li, Maobo An, Shengxiang Gao, Dan Wu, and Faru Zhao. Ensemble Learning for Countermeasure of Audio Replay Spoofing Attack in ASVspoof2017. In *Proc. Interspeech 2017*, pages 87–91, 2017. doi: 10.21437/Interspeech.2017-1246. URL http://dx.doi.org/10.21437/Interspeech.2017-1246.

Chi Jin, Yuchen Zhang, Sivaraman Balakrishnan, Martin J. Wainwright, and Michael I. Jordan. Local Maxima in the Likelihood of Gaussian Mixture Models: Structural Results and Algorithmic Consequences. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4116–4124, 2016.

Ian Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374:20150202, 04 2016. doi: 10.1098/rsta.2015.0202.

Jee-Weon Jung, Hee-Soo Heo, Il-Ho Yang, Hye-Jin Shim, and Ha-Jin Yu. A Complete End-to-End Speaker Verification System Using Deep Neural Networks: From Raw Signals to Verification Result. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5349–5353, 2018.

Jee-weon Jung, Hye jin Shim, Hee-Soo Heo, and Ha-Jin Yu. Replay Attack Detection with Complementary High-Resolution Information Using End-to-End DNN for the ASVspoof 2019 Challenge. In *Proc. Interspeech 2019*, pages 1083–1087, 2019. doi: 10.21437/Interspeech.2019-1991. URL http://dx.doi.org/10.21437/Interspeech.2019-1991.

Jee-weon Jung, Hye jin Shim, Hee-Soo Heo, and Ha-Jin Yu. A study on the role of subsidiary information in replay attack spoofing detection. *arXiv preprint arXiv:2001.11688*, 2020.

Madhu Kamble and Hemant Patil. Novel Variable Length Energy Separation Algorithm Using Instantaneous Amplitude Features for Replay Detection. In *Proc. Interspeech 2018*, pages 646–650, 2018. doi: 10.21437/Interspeech.2018-1687. URL http://dx.doi.org/10.21437/Interspeech.2018-1687.

Madhu Kamble, Hemlata Tak, and Hemant Patil. Effectiveness of Speech Demodulation-Based Features for Replay Detection. In *Proc. Interspeech*

2018, pages 641–645, 2018. doi: 10.21437/Interspeech.2018-1675. URL `http://dx.doi.org/10.21437/Interspeech.2018-1675`.

Madhu R. Kamble and Hemant A. Patil. Analysis of Reverberation via Teager Energy Features for Replay Spoof Speech Detection. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2607–2611, May 2019.

Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. ACVAE-VC: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder. *arXiv preprint arXiv:1808.05092*, 2018a.

Hirokazu Kameoka, Li Li, Shota Inoue, and Shoji Makino. Semi-blind source separation with multichannel variational autoencoder. *arXiv preprint arXiv:1808.00892*, 2018b.

Ahilan Kanagasundaram. *Speaker Verification using I-vector Features*. PhD thesis, Queensland University of Technology, October 2014.

Shachar Kaufman, Saharon Rosset, and Claudia Perlich. Leakage in Data Mining: Formulation, Detection, and Avoidance. volume 6, pages 556–563, 01 2011. doi: 10.1145/2020408.2020496.

Patrick Kenny, Vishwa Gupta, Themos Stafylakis, Pierre Ouellet, and Md Jahangir Alam. Deep neural networks for extracting baum-welch statistics for speaker recognition. In *Proc. Odyssey Speaker and Language Recognition Workshop*, 06 2014.

Elie Khoury, Tomi Kinnunen, Aleksandr Sizov, Zhizeng Wu, and Sebastian Marcel. Introducing i-vectors for joint anti-spoofing and speaker verification. In *Proc. Interspeech 2014*, pages 61–65, September 2014.

Kihyuk Sohn and Honglak Lee and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, page 3483–3491, 2015.

Taejun Kim, Jongpil Lee, and Juhan Nam. Sample-Level CNN Architectures for Music Auto-Tagging Using Raw Waveforms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 366–370, 2018.

Diederik P. Kingma and Jimmy Ba. An investigation of dependencies between frequency components and speaker characteristics for text-independent speaker identification. *Speech Communication*, 50:1820–1824, 2008.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Tomi Kinnunen and Haizhou Li. An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, 52(1): 12 – 40, 2010a. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2009.08.009. URL `http://www.sciencedirect.com/science/article/pii/S0167639309001289`.

Tomi Kinnunen and Haizhou Li. An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, 52(1):12 – 40, 2010b. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2009.08.009.

Tomi Kinnunen, Zhi-Zheng Wu, Kong Aik Lee, Filip Sedlak, Eng Siong Chng, and Haizhou Li. Vulnerability of speaker verification systems against voice conversion spoofing attacks: The case of telephone speech. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4401–4404, 2012.

Tomi Kinnunen, Mohammad Sahidullah, Hector Delgado, Massimiliano Todisco, Nicholas Evans, Junichi Yamagishi, and Kong Aik Lee. The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection. In *Proc. Interspeech 2017*, 2017a.

Tomi Kinnunen, Mohammad Sahidullah, Mauro Falcone, Luca Costantini, Rosa González Hautamäki, Dennis Thomsen, Achintya Sarkar, Zheng-Hua Tan, Héctor Delgado, Massimiliano Todisco, Nicholas Evans, Ville Hautamäki, and Kong Aik Lee. RedDots Replayed: A New Replay Spoofing Attack Corpus for Text-dependent Speaker Verification Research. In *ICASSP 2017*. IEEE, 2017b.

Tomi Kinnunen, Kong Aik Lee, Hector Delgado, Nicholas Evans, Massimiliano Todisco, Mohammad Sahidullah, Junichi Yamagishi, and Douglas A. Reynolds. t-DCF: a Detection Cost Function for the Tandem Assessment of Spoofing Countermeasures and Automatic Speaker Verification. In *Proc. Speaker Odyssey*, 2018.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. *arXiv preprint arXiv:1706.02515*, 2017.

Cheng-I Lai, Alberto Abad, Korin Richmond, Junichi Yamagishi, Najim Dehak, and Simon King. Attentive Filtering Networks for Audio Replay Attack Detection. In *Proc. ICASSP*, pages 6316–6320, May 2019a.

Cheng-I Lai, Nanxin Chen, Jesús Villalba, and Najim Dehak. ASSERT: Anti-Spoofing with Squeeze-Excitation and Residual Networks. In *Proc. Interspeech 2019*, pages 1013–1017, 2019b. doi: 10.21437/Interspeech.2019-1794. URL `http://dx.doi.org/10.21437/Interspeech.2019-1794`.

Anthony Larcher, Kong Aik Lee, Bin Ma, and Haizhou Li. Text-dependent speaker verification: Classifiers, databases and RSR2015. *Speech Communication*, 60:56 – 77, 2014. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2014.03.001.

Yee Lau, Dat Tran, and Michael Wagner. Testing Voice Mimicry with the YOHO Speaker Verification Corpus. volume 3684, pages 15–21, 09 2005. doi: 10.1007/11554028_3.

Yee Wah Lau, Michael Wagner, and Dat Tran. Vulnerability of speaker verification to voice mimicking. In *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, pages 145–148, 2004.

Galina Lavrentyeva, Sergey Novoselov, Egor Malykh, Alexander Kozlov, Kudashev Oleg, and Vadim Shchemelinin. Audio Replay Attack Detection with Deep Learning Frameworks. In *Proc. Interspeech 2017*, pages 82–86, 2017.

Galina Lavrentyeva, Sergey Novoselov, Andzhukaev Tseren, Marina Volkova, Artem Gorlanov, and Alexandr Kozlov. STC Antispoofing Systems for the ASVspoof2019 Challenge. In *Proc. Interspeech 2019*, pages 1033–1037, 2019. doi: 10.21437/Interspeech.2019-1768. URL `http://dx.doi.org/10.21437/Interspeech.2019-1768`.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–345. Springer Verlag, Jan 1999. ISBN 3540667229.

Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms. In *14th International Conference on Sound and Music Computing (SMC)*, 2017.

Kong Aik Lee, Anthony Larcher, Guangsen Wang, Patrick Kenny, Niko Brummer, David van Leeuwen, Hagai Aronowitz, Marcel Kockmann, Carlos Vaquero, Bin Ma, Haizhou Li, Themos Stafylakis, Jahangir Alam, Albert Swart,

and Javier Perez. The RedDots Data Collection for Speaker Recognition. In *Proc. Interspeech 2015*, pages 2996–3000, 2015.

Simon Leglaive, Umut Simsekli, Antoine Liutkus, Laurent Girin, and Radu Horaud. Speech enhancement with variational autoencoders and alpha-stable distributions. In *Proc. ICASSP*, pages 541–545, Brighton, United Kingdom, May 2019. IEEE.

Alexander Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, 2012. ISBN 9781118393550.

Dongbo Li, Longbiao Wang, Jianwu Dang, Meng Liu, Zeyan Oo, Seiichi Nakagawa, Haotian Guan, and Xiangang Li. Multiple Phase Information Combination for Replay Attacks Detection. In *Proc. Interspeech 2018*, pages 656–660, 2018. doi: 10.21437/Interspeech.2018-2001. URL `http://dx.doi.org/10.21437/Interspeech.2018-2001`.

Lantian Li, Yixiang Chen, Dong Wang, and Thomas Fang Zheng. A Study on Replay Attack and Anti-Spoofing for Automatic Speaker Verification. In *Proc. Interspeech 2017*, pages 92–96, 2017. doi: 10.21437/Interspeech.2017-456. URL `http://dx.doi.org/10.21437/Interspeech.2017-456`.

Li Li, Hirokazu Kameoka, and Shoji Makino. Fast MVAE: Joint Separation and Classification of Mixed Sources Based on Multichannel Variational Autoencoder with Auxiliary Classifier. In *Proc. ICASSP*, pages 546–550, May 2019a.

Rongjin Li, Miao Zhao, Zheng Li, Lin Li, and Qingyang Hong. Anti-Spoofing Speaker Verification System with Multi-Feature Integration and Multi-Task Learning. In *Proc. Interspeech 2019*, pages 1048–1052, 2019b. doi: 10.21437/Interspeech.2019-1698. URL `http://dx.doi.org/10.21437/Interspeech.2019-1698`.

Lang Lin, Rangding Wang, and Diqun Yan. A Replay Speech Detection Algorithm Based on Sub-band Analysis. In *10th IFIP TC 12 International Conference, IIP 2018, Nanning, China, October 19-22, 2018, Proceedings*, pages 337–345, 01 2018.

Johan Lindberg and Mats Blomberg. Vulnerability in speaker verification - a study of technical impostor techniques. In *Proc. European Conference on Speech Communication and Technology*, volume 3, page 1211–1214, 01 1999.

Zhang Lipeng, Cao Jiang, Xu Mingxing, and Zheng Fang. Prevention of impostors entering speaker recognition systems. *Journal of Tsinghua University*, 48:699–703, 04 2008.

Meng Liu, Longbiao Wang, Jianwu Dang, Seiichi Nakagawa, Haotian Guan, and Xiangang Li. Replay Attack Detection Using Magnitude and Phase Information with Attention-based Adaptive Filters. In *Proc. ICASSP*, pages 6201–6205, May 2019a.

Songxiang Liu, Haibin Wu, Hung yi Lee, and Helen Meng. Adversarial Attacks on Spoofing Countermeasures of Automatic Speaker Verification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 312–319, 2019b.

Saranya M S and Hema Murthy. Decision-level Feature Switching as a Paradigm for Replay Attack Detection. In *Proc. Interspeech 2018*, pages 686–690, September 2018. doi: 10.21437/Interspeech.2018-1494. URL `http://dx.doi.org/10.21437/Interspeech.2018-1494`.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of ICML*, volume 30, 06 2013.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 1:1–48, 2008.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. pages 5188–5196, 06 2015. doi: 10.1109/CVPR.2015.7299155.

Johnny Mariéthoz and Samy Bengio. Can a Professional Imitator Fool a GMM-Based Speaker Verification System? *Idiap Research Report Number: Idiap-RR-61-2005*, 01 2005.

Takashi Masuko, Takafumi Hitotsumatsu, Keiichi Tokuda, and Takao Kobayashi. On The Security of Hmm-Based Speaker Verification Systems Against Imposture Using Synthetic Speech. In *In Proceedings of the European Conference on Speech Communication and Technology*, pages 1223–1226, 1999.

Driss Matrouf, Jean francois Bonastre, and Corinne Fredouille. Effect of Speech Transformation on Impostor Acceptance. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I, 2006.

Alex F. Mendelson, Maria A. Zuluaga, Marco Lorenzi, Brian F. Hutton, and Sébastien Ourselin. Selection bias in the reported performances of AD classification pipelines. *NeuroImage: Clinical*, 14:400 – 416, 2017. ISSN 2213-1582. doi: https://doi.org/10.1016/j.nicl.2016.12.018. URL `http://www.sciencedirect.com/science/article/pii/S221315821630256X`.

Jia Min Karen Kua, Tharmarajah Thiruvaran, Mohaddeseh Nosratighods, Eliathamby Ambikairajah, and Julien Epps. Investigation of spectral centroid magnitude and frequency for speaker recognition. In *Proc. Odyssey Speaker and Language Recognition Workshop*, pages 34–39, 2010.

Saumitra Mishra. *Interpretable Machine Learning for Machine Listening*. PhD thesis, Queen Mary University of London, May 2020.

Saumitra Mishra, Bob L. Sturm, and Simon Dixon. Local Interpretable Model-Agnostic Explanations for Music Content Analysis. In *Proc. ISMIR*, 2017.

Saumitra Mishra, Bob L. Sturm, and Simon Dixon. Understanding a Deep Machine Listening Model Through Feature Inversion. In *Proc. ISMIR*, 2018.

Saumitra Mishra, Daniel Stoller, Emmanouil Benetos, Bob L. Sturm, and Simon Dixon. Gan-based Generation and Automatic Selection of Explanations for Neural Networks. *arXiv preprint arXiv:1904.09533*, 2019.

Seyed Hamidreza Mohammadi and Alexander Kain. An overview of voice conversion systems. *Speech Communication*, 88:65 – 82, 2017. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2017.01.008. URL `http://www.sciencedirect.com/science/article/pii/S0167639315300698`.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for Interpreting and Understanding Deep Neural Networks. *Digital Signal Processing*, 73:1–15, 2018.

Veronica Morfi and Dan Stowell. Deep Learning for Audio Event Detection and Tagging on Low-Resource Datasets. *Applied Sciences*, 8(8):1397, 2018. ISSN 2076-3417. doi: 10.3390/app8081397. URL `http://dx.doi.org/10.3390/app8081397`.

Masanori Morise, Fumiya YOKOMORI, and Kenji OZAWA. WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications. *IEICE Transactions on Information and Systems*, E99.D(7):1877–1884, 2016. doi: 10.1587/transinf.2015EDP7457.

Hannah Muckenhirn, Pavel Korshunov, Mathew Magimai-Doss, and Sébastien Marcel. Long-Term Spectral Statistics for Voice Presentation Attack Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(11):2098–2111, 2017a. ISSN 2329-9290. doi: 10.1109/TASLP.2017.2743340.

Hannah Muckenhirn, Mathew Magimai-Doss, and Sébastien Marcel. End-to-End convolutional neural network-based voice presentation attack detection. In *IEEE International Joint Conference on Biometrics (IJCB)*, pages 335–341, Oct 2017b.

Hannah Muckenhirn, Mathew Magimai Doss, and Sébastien Marcell. Towards Directly Modeling Raw Speech Signal for Speaker Verification Using CNNS. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4884–4888, 2018.

Parav Nagarsheth, Elie Khoury, Kailash Patil, and Matt Garland. Replay Attack Detection Using DNN for Channel Discrimination. In *Proc. Interspeech 2017*, pages 97–101, 2017.

Vinod Nair and Geoffrey Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of ICML*, volume 27, pages 807–814, 06 2010.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

Sergey Novoselov, Alexandr Kozlov, Galina Lavrentyeva, Konstantin Simonchik, and Vadim Shchemelinin. STC anti-spoofing systems for the ASVspoof 2015 challenge. In *Proc. ICASSP*, pages 5475–5479, March 2016a.

Sergey Novoselov, Alexandr Kozlov, Galina Lavrentyeva, Konstantin Simonchik, and Vadim Shchemelinin. STC Anti-spoofing Systems for the ASVspoof 2015 Challenge. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5475–5479, 2016b.

Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, Ioannis Kompatsiaris, Katharina Kinder-Kurlanda, Claudia Wagner, Fariba Karimi, Miriam Fernandez, Harith Alani, Bettina Berendt, Tina Kruegel, Christian Heinze, Klaus Broelemann, Gjergji Kasneci, Thanassis Tiropanis, and Steffen Staab. Bias in data-driven

artificial intelligence systems—An introductory survey. *WIREs Data Mining and Knowledge Discovery*, 10, June 2020.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499*, 2016.

Tanvina B. Patel and Hemant A. Patil. Combining evidences from mel cepstral, cochlear filter cepstral and instantaneous frequency features for detection of natural vs. spoofed speech. In *Proc. Interspeech 2015*, pages 2062–2066, 2015.

Ankur T. Patil, Rajul Acharya, Pulikonda Aditya Sai, and Hemant A. Patil. Energy Separation-Based Instantaneous Frequency Estimation for Cochlear Cepstral Feature for Replay Spoof Detection. In *Proc. Interspeech 2019*, pages 2898–2902, 2019. doi: 10.21437/Interspeech.2019-2742. URL `http://dx.doi.org/10.21437/Interspeech.2019-2742`.

Hemant A. Patil, Madhu R. Kamble, Tanvina B. Patel, and Meet Soni. Novel Variable Length Teager Energy Separation Based Instantaneous Frequency Features for Replay Detection. In *Proc. Interspeech 2017*, 2017.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Bryan L. Pellom and John H. L. Hansen. An experimental study of speaker verification sensitivity to computer voice-altered imposters. In *Proc. ICASSP*, pages 837–840, Mar 1999.

Patrick Perrot, Guido Aversano, and Gerard Chollet. Voice Disguise and Automatic Detection: Review and Perspectives. In *Progress in Nonlinear Speech Processing. Lecture Notes in Computer Science, vol 4391. Springer, Berlin, Heidelberg*, pages 101–117, 01 2005. doi: 10.1007/978-3-540-71505-4_7.

Patrick von Platen, Fei Tao, and Gokhan Tur. Multi-Task Siamese Neural Network for Improving Replay Attack Detection. *arXiv preprint arXiv:2002.07629*, 2020.

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of

images, labels and captions. In *Advances in neural information processing systems*, page 2352–2360, 2016.

Yanmin Qian, Nanxin Chen, and Kai Yu. Deep features for automatic spoofing detection. *Speech Communication*, 85:43 – 52, 2016. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2016.10.007. URL `http://www.sciencedirect.com/science/article/pii/S0167639316301091`.

Mirco Ravanelli and Yoshua Bengio. Speaker Recognition from Raw Waveform with SincNet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028, 2018.

Douglas A Reynolds. Speaker identification and verification using Gaussian mixture speaker models. *Speech communication*, 17(1):91–108, 1995.

F. A. Rezaur rahman Chowdhury, Quan Wang, Ignacio Lopez Moreno, and Li Wan. Attention-Based Models for Text-Dependent Speaker Verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5359–5363, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *Proc. Knowledge Discovery and Data Mining(KDD)*, 2016.

Francisco Rodríguez-Algarra, Bob Sturm, and Simon Dixon. Characterising Confounding Effects in Music Classification Experiments through Interventions. *Transactions of the International Society for Music Information Retrieval*, 2:52–66, 08 2019. doi: 10.5334/tismir.24.

Saharon Rosset, Claudia Perlich, Grzegorz Swirszcz, Prem Melville, and Yan Liu. Medical data mining: Insights from winning two competitions. *Data Mining and Knowledge Discovery*, 20:439–468, 05 2010. doi: 10.1007/s10618-009-0158-x.

Evgenia Rusak, Lukas Schott, Roland S. Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. Increasing the robustness of DNNs against image corruptions by playing the Game of Noise. *arXiv preprint arXiv:2001.06057*, 2020.

Seyed Omid Sadjadi, Malcolm Slaney, , and Larry Heck. MSR Identity Toolbox v1.0: A MATLAB Toolbox for Speaker Recognition Research. *Speech and Language Processing Technical Committee Newsletter*, 2013.

Mohammad. Sahidullah, Tomi Kinnunen, and Cemal Hanilçi. A Comparison of Features for Synthetic Speech Detection. In *Proc. Interspeech 2015*, pages 2087–2091, 2015.

Mohammad Sahidullah, Ḥector Delgado, Massimiliano Todisco, Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, and Kong-Aik Lee. Introduction to Voice Presentation Attack Detection and Recent Advances, 2019.

Hardik Sailor, Madhu Kamble, and Hemant Patil. Auditory Filterbank Learning for Temporal Modulation Features in Replay Spoof Speech Detection. In *Proc. Interspeech 2018*, pages 666–670, 2018. doi: 10.21437/Interspeech.2018-1651. URL `http://dx.doi.org/10.21437/Interspeech.2018-1651`.

Justin Salamon, Juan Pablo Bello, Andrew Farnsworth, and Steve Kelling. Fusing shallow and deep learning for bioacoustic bird species classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 141–145, 2017.

Sai Samarth R Phaye, Emmanouil Benetos, and Ye Wang. SubSpectralNet – Using Sub-spectrogram Based Convolutional Neural Networks for Acoustic Scene Classification. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 825–829, May 2019.

Wei Shang and Maryhelen Stevenson. Score normalization in playback attack detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1678–1681, 2010.

Hye-jin Shim, Hee-Soo Heo, Jee weon Jung, and Ha-Jin Yu. Self-supervised pre-training with acoustic configurations for replay spoofing detection. *arXiv preprint arXiv:1910.09778*, 2019.

David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. Deep Neural Network Embeddings for Text-Independent Speaker Verification. pages 999–1003, 08 2017. doi: 10.21437/Interspeech.2017-620.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, 2018.

Meet H. Soni, Tanvina B. Patel, and Hemant A. Patil. Novel subband autoencoder features for detection of spoofed speech. In *Proc. Interspeech 2016*, pages 1820–1824, 2016. doi: 10.21437/Interspeech.2016-668. URL `http://dx.doi.org/10.21437/Interspeech.2016-668`.

Kaavya Sriskandaraja, Vidhyasaharan Sethu, Phu Ngoc Le, and Eliathamby Ambikairajah. Investigation of Sub-Band Discriminative Information Between

Spoofed and Genuine Speech. In *Proc. Interspeech 2016*, pages 1710–1714, 2016.

Kaavya Sriskandaraja, Vidhyasaharan Sethu, and Eliathamby Ambikairajah. Deep Siamese Architecture Based Replay Detection for Secure Voice Biometric. In *Proc. Interspeech 2018*, pages 671–675, 2018. doi: 10.21437/Interspeech.2018-1819. URL `http://dx.doi.org/10.21437/Interspeech.2018-1819`.

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-U-Net: A Multi-Scale Neural Network for End-to-End Source Separation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 19, pages 334–340, 2018.

Dan Stowell, Tereza Petruskova, Martin Šálek, and Pavel Linhart. Automatic acoustic identification of individuals in multiple species: improving identification across recording conditions. *Journal of The Royal Society Interface*, 16, 04 2019. doi: 10.1098/rsif.2018.0940.

Bob L. Sturm. The GTZAN Dataset: Its Contents, its Faults, their Effects on Evalu- ation, and its Future Use. *arXiv preprint arXiv:1306.1461*, 2013.

Bob L. Sturm. A Simple Method to Determine if a Music Information Retrieval System is a "Horse". *IEEE Transactions on Multimedia*, 16(6):1636–1644, Oct 2014.

Bob L. Sturm. The "Horse" Inside: Seeking Causes Behind the Behaviours of Music Content Analysis Systems. *arXiv preprint arXiv:1606.03044*, 2016.

Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Chris Pal, and Aaron Courville. Adversarial Generation of Natural Language. In *Proceedings of the 2^{nd} Workshop on Representation Learning for NLP*, pages 241–251, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2629.

Gajan Suthokumar, Vidhyasaharan Sethu, Chamith Wijenayake, and Eliathamby Ambikairajah. Modulation Dynamic Features for the Detection of Replay Attacks. In *Proc. Interspeech 2018*, pages 691–695, September 2018.

Gajan Suthokumar, Kaavya Sriskandaraja, Vidhyasaharan Sethu, Chamith Wijenayake, and Eliathamby Ambikairajah. Phoneme Specific Modelling and Scoring Techniques for Anti Spoofing System. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6106–6110, May 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL `http://arxiv.org/abs/1312.6199`.

Dávid Sztahó, György Szaszák, and András Beke. Deep learning methods in speaker recognition: a review. *arXiv preprint arXiv:1911.06615*, 2019.

Hemlata Tak, Jose Patino, Andreas Nautsch, Nicholas Evans, and Massimiliano Todisco. An explainability study of the constant Q cepstral coefficient spoofing countermeasure for automatic speaker verification. *arXiv preprint arXiv:2004.06422*, 2020.

Shawn Tan and Khe Chai Sim. Learning utterance-level normalisation using Variational Autoencoders for robust automatic speech recognition. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 43–49, Dec 2016.

Zheng-Hua Tan, Achintya kr. Sarkar, and Najim Dehak. rVAD: An unsupervised segment-based robust voice activity detection method. *Computer Speech & Language*, 59:1 – 21, 2020. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2019.06.005.

Yun Tang, Guohong Ding, Jing Huang, Xiaodong He, and Bowen Zhou. Deep Speaker Embedding Learning with Multi-level Pooling for Text-independent Speaker Verification. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6116–6120, 2019.

Paul Taylor. *Text-to-Speech Synthesis*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521899273.

Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Aboul Ella Hassanien. Linear discriminant analysis: A detailed tutorial. *AI Communications*, 30: 169–190,, 05 2017. doi: 10.3233/AIC-170729.

Massimiliano Todisco, Hector Delgado, and Nicholas Evans. A new feature for automatic speaker verification anti-spoofing: Constant Q cepstral coefficients. In *Proc. Odyssey Speaker and Language Recognition Workshop*, pages 283–290, June 2016.

Massimiliano Todisco, Ḧector Delgado, and Nicholas Evans. Constant Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. *Computer Speech and Language,*, Volume 45,:Pages 516–535, 2017.

Massimiliano Todisco, Xin Wang, Ville Vestman, Mohammad Sahidullah, Ḧector Delgado, Andreas Nautsh, Junichi Yamagishi, Nicholas Evans, Tomi

Kinnunen, and Kong Aik Lee. ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection. In *Proc. Interspeech 2019*, 2019.

Francis Tom, Mohit Jain, and Prasenjit Dey. End-To-End Audio Replay Attack Detection Using Deep Convolutional Networks with Attention. In *Proc. Interspeech 2018*, pages 681–685, 2018.

Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A Deeper Look at Dataset Bias. *arXiv preprint arXiv:1505.01257*, 2015.

Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528, 2011.

Youzhi Tu, Man-Wai Mak, and Jen-Tzung Chien. Variational Domain Adversarial Learning for Speaker Verification. In *Proc. Interspeech 2019*, pages 4315–4319, 2019. doi: 10.21437/Interspeech.2019-2168. URL `http://dx.doi.org/10.21437/Interspeech.2019-2168`.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing Motion and Content for Video Generation. *arXiv preprint arXiv:1707.04993*, 2017.

Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4052–4056, 2014.

Jesús Villalba and Eduardo Lleida. Speaker verification performance degradation against spoofing and tampering attacks. In *FALA 2010*, pages 131–134, 2010.

Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. *arXiv preprint arXiv:1606.07873*, 2016.

Hongji Wang, Heinrich Dinkel, Shuai Wang, Yanmin Qian, and Kai Yu. Cross-Domain Replay Spoofing Attack Detection Using Domain Adversarial Training. In *Proc. Interspeech 2019*, pages 2938–2942, 2019a. doi: 10.21437/Interspeech.2019-2120. URL `http://dx.doi.org/10.21437/Interspeech.2019-2120`.

Xianliang Wang, Yanhong Xiao, and Xuan Zhu. Feature Selection Based on CQCCs for Automatic Speaker Verification Spoofing. In *Proc. Interspeech 2017*, pages 32–36, 2017. doi: 10.21437/Interspeech.2017-304. URL `http://dx.doi.org/10.21437/Interspeech.2017-304`.

Xin Wang, Junichi Yamagishi, Massimiliano Todisco, Hector Delgado, Andreas Nautsch, Nicholas Evans, Mohammad Sahidullah, Ville Vestman, Tomi Kinnunen, Kong Aik Lee, Lauri Juvela, Paavo Alku, Yu-Huai Peng, Hsin-Te Hwang, Yu Tsao, Hsin-Min Wang, Sebastien Le Maguer, Markus Becker, Fergus Henderson, Rob Clark, Yu Zhang, Quan Wang, Ye Jia, Kai Onuma, Koji Mushika, Takashi Kaneda, Yuan Jiang, Li-Juan Liu, Yi-Chiao Wu, Wen-Chin Huang, Tomoki Toda, Kou Tanaka, Hirokazu Kameoka, Ingmar Steiner, Driss Matrouf, Jean-Francois Bonastre, Avashna Govender, Srikanth Ronanki, Jing-Xuan Zhang, and Zhen-Hua Ling. ASVspoof 2019: a large-scale public database of synthetic, converted and replayed speech. *arXiv preprint arXiv:1911.01601*, 2019b.

Zhi-Feng Wang, Gang Wei, and Qian-Hua He. Channel pattern noise based playback attack detection algorithm for speaker recognition. In *Proc. International Conference on Machine Learning and Cybernetics*, volume 4, page 1708–1713, July 2011.

Buddhi Wickramasinghe, Saad Irtza, Eliathamby Ambikairajah, and Julien Epps. Frequency Domain Linear Prediction Features for Replay Spoofing Attack Detection. In *Proc. Interspeech 2018*, pages 661–665, 2018. doi: 10.21437/Interspeech.2018-1574. URL `http://dx.doi.org/10.21437/Interspeech.2018-1574`.

Buddhi Wickramasinghe, Eliathamby Ambikairajah, and Julien Epps. Biologically Inspired Adaptive-Q Filterbanks for Replay Spoofing Attack Detection. In *Proc. Interspeech 2019*, pages 2953–2957, 2019a. doi: 10.21437/Interspeech.2019-1535. URL `http://dx.doi.org/10.21437/Interspeech.2019-1535`.

Buddhi Wickramasinghe, Eliathamby Ambikairajah, Julien Epps, Vidhyasaharan Sethu, and Haizhou Li. Auditory Inspired Spatial Differentiation for Replay Spoofing Attack Detection. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6011–6015, May 2019b.

Jennifer Williams and Joanna Rownicka. Speech Replay Detection with x-Vector Attack Embeddings and Spectral Features. In *Proc. Interspeech 2019*, pages 1053–1057, 2019. doi: 10.21437/Interspeech.2019-1760. URL `http://dx.doi.org/10.21437/Interspeech.2019-1760`.

Marcin Witkowski, Stanisław Kacprzak, Piotr Żelasko, Konrad Kowalczyk, and Jakub Gałka. Audio Replay Attack Detection Using High-Frequency Features. In *Proc. Interspeech 2017*, pages 27–31, 2017.

Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A Light CNN for Deep Face Representation with Noisy Labels. *arXiv preprint arXiv:1511.02683*, 2015a.

Zhanghao Wu, Shuai Wang, Yanmin Qian, and Kai Yu. Data Augmentation Using Variational Autoencoder for Embedding Based Speaker Verification. In *Proc. Interspeech 2019*, pages 1163–1167, 2019. doi: 10.21437/Interspeech.2019-2248. URL `http://dx.doi.org/10.21437/Interspeech.2019-2248`.

Zhizheng Wu, Sheng Gao, Eng Siong Cling, and Haizhou Li. A study on replay attack and anti-spoofing for text-dependent speaker verification. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–5, Dec 2014a.

Zhizheng Wu, Sheng Gao, Eng Siong Cling, and Haizhou Li. A study on replay attack and anti-spoofing for text-dependent speaker verification. In *Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA)*, pages 1–5. IEEE, 2014b.

Zhizheng Wu, Nicholas Evans, Tomi Kinnunen, Junichi Yamagishi, Federico Alegre, and Haizhou Li. Spoofing and countermeasures for speaker verification: A survey. *Speech Communication*, 66:130 – 153, 2015b. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2014.10.005. URL `http://www.sciencedirect.com/science/article/pii/S0167639314000788`.

Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Cemal Hanilci, Mohammad Sahidullah, and Aleksandr Sizov. ASVspoof 2015: the First Automatic Speaker Verification Spoofing and Countermeasures Challenge. In *Proc. Interspeech 2015*, 2015c.

Zhizheng Wu, Junichi Yamagishi, Tomi Kinnunen, Cemal Hanilçi, Mohammed Sahidullah, Aleksandr Sizov, Nicholas Evans, Massimiliano Todisco, and Héctor Delgado. ASVspoof: The Automatic Speaker Verification Spoofing and Countermeasures Challenge. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):588–604, June 2017. ISSN 1932-4553.

Sibel Yaman, Jason Pelecanos, and Ruhi Sarikaya. Bottleneck Features for Speaker Recognition. In *Proc. Odyssey Speaker and Language Recognition Workshop*, 01 2012.

Jichen Yang, Rohan Kumar Das, and Haizhou Li. Extended Constant-Q Cepstral Coefficients for Detection of Spoofing Attacks. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1024–1029, Nov 2018a.

Jichen Yang, Changhuai You, and Qianhua He. Feature with Complementarity of Statistics and Principal Information for Spoofing Detection. In *Proc. Interspeech 2018*, pages 651–655, 2018b. doi: 10.21437/Interspeech.2018-1693. URL http://dx.doi.org/10.21437/Interspeech.2018-1693.

Jichen Yang, Rohan Kumar Das, and Nina Zhou. Extraction of Octave Spectra Information for Spoofing Attack Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2373–2384, 2019a.

Yexin Yang, Hongji Wang, Heinrich Dinkel, Zhengyang Chen, Shuai Wang, Yanmin Qian, and Kai Yu. The SJTU Robust Anti-Spoofing System for the ASVspoof 2019 Challenge. In *Proc. Interspeech 2019*, pages 1038–1042, 2019b. doi: 10.21437/Interspeech.2019-2170. URL http://dx.doi.org/10.21437/Interspeech.2019-2170.

Chang Huai You, Jichen Yang, and Huy Dat Tran. Device Feature Extractor for Replay Spoofing Detection. In *Proc. Interspeech 2019*, pages 2933–2937, 2019. doi: 10.21437/Interspeech.2019-2137. URL http://dx.doi.org/10.21437/Interspeech.2019-2137.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *arXiv preprint arXiv:1712.07107*, 2017.

Hossein Zeinali, Themos Stafylakis, Georgia Athanasopoulou, Johan Rohdin, Ioannis Gkinis, Lukáš Burget, and Jan Černocký. Detecting Spoofing Attacks Using VGG and SincNet: BUT-Omilia Submission to ASVspoof 2019 Challenge. In *Proc. Interspeech 2019*, pages 1073–1077, 2019. doi: 10.21437/Interspeech.2019-2892. URL http://dx.doi.org/10.21437/Interspeech.2019-2892.

Heiga Zen, Keiichi Tokuda, and Alan W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039 – 1064, 2009. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2009.04.004. URL http://www.sciencedirect.com/science/article/pii/S0167639309000648.

Elisabeth Zetterholm. Detection of Speaker Characteristics Using Voice Imitation. pages 192–205, 01 2007. doi: 10.1007/978-3-540-74122-0_16.

Chunlei Zhang, Chengzhu Yu, and John HL Hansen. An Investigation of Deep-Learning Frameworks for Speaker Verification Antispoofing. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):684–694, 2017.

Ya-Jie Zhang, Shifeng Pan, Lei He, and Zhen-Hua Ling. Learning Latent Representations for Style Control and Transfer in End-to-end Speech Synthesis. In *Proc. ICASSP*, pages 6945–6949, May 2019a.

Yang Zhang, Lantian Li, and Dong Wang. VAE-Based Regularization for Deep Speaker Embedding. In *Proc. Interspeech 2019*, pages 4020–4024, 2019b. doi: 10.21437/Interspeech.2019-2486. URL `http://dx.doi.org/10.21437/Interspeech.2019-2486`.