Vérification automatique du locuteur robuste à l'usurpation de l'identité : Architecture, Explicabilité et Optimisation Conjointe

Wanying Ge

Spoofing-robust Automatic Speaker Verification: Architecture, Explainability and Joint Optimisation

Dissertation

submitted to

Sorbonne Université

in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Author:
Wanying Ge

Defended on the 30^{th} *of May* 2024
before a committee composed of:

Reviewers	**Prof. Marie Tahon**, Le Mans Université, France
	Prof. Driss Matrouf, Avignon Université, France
President	**Prof. Marie Tahon**, Le Mans Université, France
Examiners	**Asst. Prof. Chiara Galdi**, EURECOM, France
	Prof. Kong Aik Lee, The Hong Kong Polytechnic University, Hong Kong SAR, China
	Prof. Itshak Lapidot, The Afeka Academic College of Engineering in Tel Aviv, Israel
Thesis advisor	**Prof. Nicholas Evans**, EURECOM, France
Co-supervisor	**Asst. Prof. Massimiliano Todisco**, EURECOM, France

Acknowledgements

I would like to thank my supervisor Prof. Nicholas Evans for his guidance on my research. Talking with him has always been encouraging and relaxing since the first interview. And I would like to thank my co-supervisor Prof. Massimiliano Todisco, for his guidance and his companionship through many conferences.

I would like to thank other group members. To Jose Patino, for his many help and advice. To Hemlata Tak, for being a nice senior and a great friend. To Michele Panariello, I'm happy that we have you in our group. To Oubaïda Chouchane, for the grilled salad she made. To Madhu Kamble, for the joy she brought into the group.

I would like to thank my family. To my dad, he once told me that he thought I could do some research related job, that was long before I decided to apply for one myself. To my mom, for when we grew up, she made us breakfast everyday even after night shifts. To my brother, us playing It Takes Two across the English Channel is one of my best memories.

The journey of this PhD began when I first introduced myself at the office. It feels like only seconds later that we got the acceptance for my first ever ICASSP paper, and just another moment later, I already started to write this thesis. It's a fortune to meet the audio group and call them my friends, and there's no better feeling than being told I was missed by the group when I was away. At the end of this journey and the beginning of many new ones, personally, nothing suits more than the lines that I am long keen to quote:

I will not forget one line of this, not one day, I swear. I will always remember when the Doctor was me.[1]

Nice, May 2024 ***Wanying Ge***

[1]The Time of the Doctor (2013)

Abstract

Voice biometric systems, particularly Automatic Speaker Verification (ASV) systems, have become indispensable technologies for providing secure, efficient, and convenient authentication. However, these systems are vulnerable to the increasingly advanced and accessible spoofing attacks facilitated by advancements in deep learning and artificial intelligence. To combat these threats, spoofing countermeasures (CMs) are designed to differentiate between genuine and spoofed voices. Despite significant performance improvements, there remains a gap in understanding how and why these CMs function, and how they interact with and complement ASV systems. This thesis aims to address these challenges and paves the way for an interpretable and integrated speaker verification anti-spoofing system.

The first contribution of this thesis includes the use of a neural architecture search (NAS) algorithm, namely Partially Connected Differentiable Architecture Search (PC-DARTS), to automatically discover optimal network architectures for voice anti-spoofing. The resulting models demonstrate competitive performance and showcase the potential of our automatically learned network. Building on the solid foundation of the application of PC-DARTS in anti-spoofing, we then apply the searching procedure in a fully end-to-end manner. Both network architecture *and* feature pre-processing operations are jointly optimised, with the algorithm operating on raw waveform input. We demonstrate that the resulting end-to-end algorithm, namely Raw PC-DARTS, is competitive with state-of-the-art solutions and shows better generalisation to unseen spoofing attacks compared to CMs that use hand-crafted features.

Though the previous work shed light on which network component and the corresponding feature representations are beneficial for spoofing detection, we still lack an explanation on *how* such design choice influences the system performance. The second contribution of this thesis involves the use of SHapley Additive exPlanations (SHAP) to study the behaviours of different state-of-the-art systems and various input features. SHAP is used to explain the contribution of each individual input feature to the model's output, and the obtained results can be visualised using heat maps of the same dimension as the input feature. We demonstrate the application of SHAP across two separate studies. We first focused on *classifier behaviour*, where we highlighted that different CM classifiers uses different spectro-temporal intervals for spoofing detection. We then shifted our focus to *attack analysis*, where we revealed the spoofing artefacts of each attack using different representations and classifiers. We demonstrate that SHAP visualisation results can be used to locate attack-specific characteristics as well as the differences and consistencies between synthetic speech and converted voice spoofing attacks.

The final contribution includes development of an integrated spoofing-aware speaker verification (SASV) system. We first present our findings that deep learning's role in both spoofing and detection exhibits variability in performance, suggesting that changing training parameters could allow attacks to bypass detection. We then present our work on enhancing CMs with ASV systems to improve detection capabilities. We demonstrate the potential of joint optimisation of ASV and CM for the SASV task. Our findings indicate that joint optimisation is successful in improving system robustness to spoofing attacks by making complementary use of the enrolment utterance of the ASV system. Although joint optimisation results in overfitting to known speakers, we found that using auxiliary data collected from new speakers can mitigate this overfitting. We also found that joint optimisation degrades the performance of individual ASV and CM sub-systems but improves their complementarity.

Contents

List of Figures

List of Tables

List of Abbreviations

ASV	Automatic Speaker Verification
CM	Countermeasure
TTS	Text-to-Speech Synthesis
VC	Voice Conversion
HMM	Hidden Markov Models
GMM	Gaussian Mixture Model
DNN	Deep Neural Network
CNN	Convolutional Neural Network
LA	Logical Access
PA	Physical Access
POI	Person of Interest
UBM	Universal Background Model
LFCC	Linear Frequency Cepstral Coefficient
CQCC	Constant Q Cepstral Coefficient
MFCC	Mel Frequency Cepstrum Coefficient
Grad-CAM	Gradient-weighted Class Activation Mapping
LIME	Local Interpretable Model-agnostic Explanations
SHAP	SHapley Additive exPlanations
LSTM	Long Short-Term Memory
NEAT	Neuro-Evolution for Augmenting Topologies
NAS	Neural Architecture Search
DARTS	Differentiable ARchiTecture Search
EER	Equal Error Rate
FAR	False Acceptance Rate
FRR	False Rejection Rate

t-DCF	Tandem Detection Cost Function
VAE	Variational Auto-Encoder
VITS	Variational Inference with adversarial learning for end-to-end Text-to-Speech
GRU	Gated Recurrent Unit
SSL	Self-Supervised leaning

Chapter 1

Introduction

The topic of this thesis is the development of effective and interpretable spoofing countermeasures, along with their integration with Automatic Speaker Verification systems, to create a more robust and reliable authentication system. While the field of speaker recognition has made significant strides in developing robust models over the years, it's only in recent times that the vulnerability of ASV systems to spoofing attacks has come to the forefront. This heightened attention can be attributed to advancements in deep learning since the 2010s, which have not only elevated the study of artificially generated speeches but also underscored the potential risks they pose to ASV systems. In response to these challenges, it is important to develop systems that can not only identify and neutralize these threats but also provide clarity on the mechanisms behind spoofing countermeasures.

This chapter first introduces the tasks at hand, i.e., speaker verification in Sec. 1.1 and spoofing detection in Sec. 1.2. More specific, motivations and aims of our work are introduced in Sec. 1.3, and the contribution of this thesis is concluded in Sec. 1.4. Finally, structure of the thesis is described in Sec. 1.5.

1.1 Automatic speaker verification

Automatic Speaker Verification (ASV) systems [4] are biometric systems designed to authenticate a speaker's claimed identity based on recordings of his or her voice. They use estimates of physiological and behavioural voice characteristics - including the vocal tract, nasal passage, lung air pressure, and intonation [5] - to model the voice and to distinguish between individuals.

An ASV system typically involves two main stages: training and test, as shown in Figure 1.1. The training stage, shown in the upper part of Figure 1.1, can be viewed as a multi-class classification problem where a speaker label is assigned to each training utterance from a set of known speakers. Classifiers such as Time Delay Neural Networks (TDNNs) [6, 7] are trained to extract features which capture speaker-specific characteristics from the speech signal. They capture acoustic patterns within the spectral envelope of short frames of the speech signal. These features are then processed through a pooling layer, which aggregates the frame-level features into utterance-level features which encapsulate the temporal dynamics of the speech signal [7, 8]. Utterance-level features are then mapped into a high dimensional space to obtain speaker embeddings which serve as a compact representation of the speaker's voice identity used in the test stage. A loss function is employed for the optimisation of the network parameters so that distances between the learned representations corresponding to the same speakers are minimised, while those corresponding to different speakers are maximised [9, 10].

The test stage, shown in the lower part of Figure 1.1, involves the comparison of a test utterance with an enrolment utterance. Both utterances are processed using the same feature extractor as in the previous stage to produce speaker embeddings. A back-end is then used to evaluate the similarity of these embeddings, to generate a decision score and subsequently to verify the claimed identity. A high score prompts the system to accept the identity claim, indicating that both utterances originate from the same speaker. Conversely, a low score leads the system to reject the claim, indicating that the pair of utterances come from different speakers [11].

1.2 Deepfake and spoofing detection

The literature shows that ASV systems are vulnerable to various types of spoofing attacks [12], raising significant security concerns. Spoofing attacks typically involve mimicking the voice characteristics of a targeted speaker. This can be done simply through the replaying of a recorded utterance [13], or by other more technical methods such as text-to-speech (TTS) synthesis or voice conversion (VC) [14, 15]. TTS and VC technologies are designed to generate artificial speech in the voice of a target speaker. Such technologies have advanced to such an extent that they can now generate realistic speech output which is indistinguishable from real human

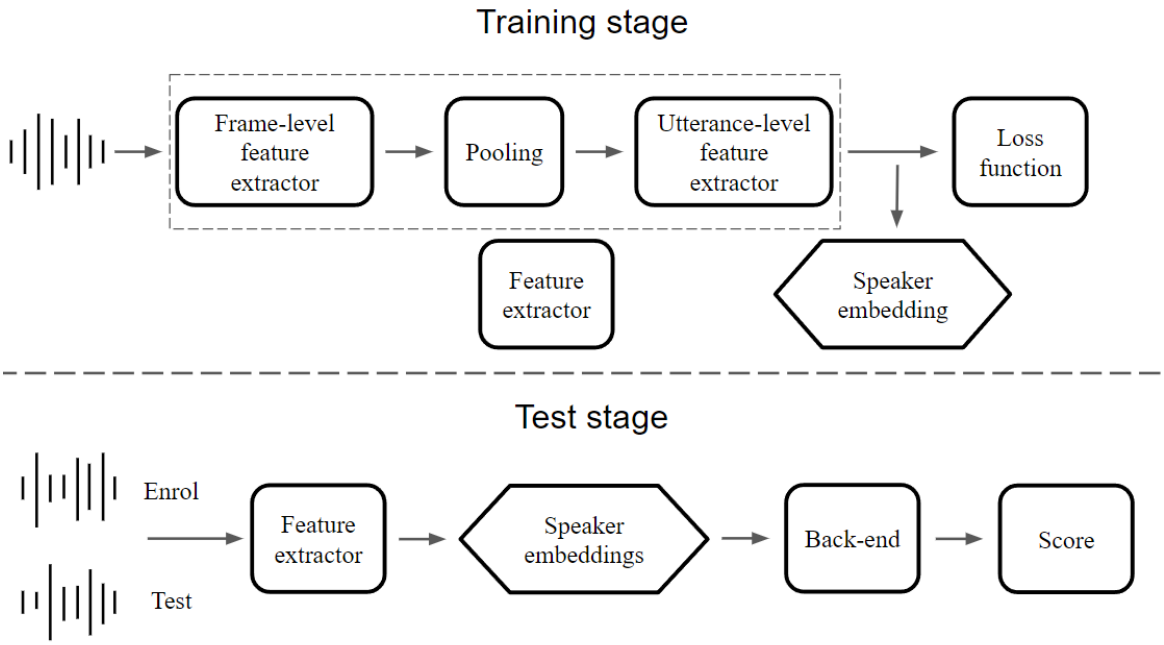


Figure 1.1: Illustration of the training and testing stages of an Automatic Speaker Verification (ASV) system. During the training stage, a feature extractor is guided by the loss function to extract discriminative speaker embeddings. In the testing stage, this same feature extractor extracts embeddings from both the enrolment and test utterances. Subsequently, a back-end system evaluates the similarity between these two embeddings, producing the final score. Figure reproduced from [1].

speeches. Since the databases used for the training of ASV systems typically only contain recordings of bona fide (genuine) speech, these systems can be vulnerable to well-crafted spoofing attacks, leading to increment in the false alarm rate [12]. Thus, there is a need for effective countermeasures (CM) against spoofing attacks to enhance the robustness and reliability of ASV systems.

Like ASV, the anti-spoofing task is a binary classification task, where a CM is used to determine whether a given speech recording is bona fide or spoofed. This task is addressed by identifying artefacts or cues left by spoofing algorithms. While some of these artefacts may be readily apparent to the human ear in the case of unsuccessful or unrealistic artificially generated recordings, many are subtle nuances beyond human perception. However, countermeasures, typically deep neural networks, are trained to detect such artefacts. Unlike ASV, the primary

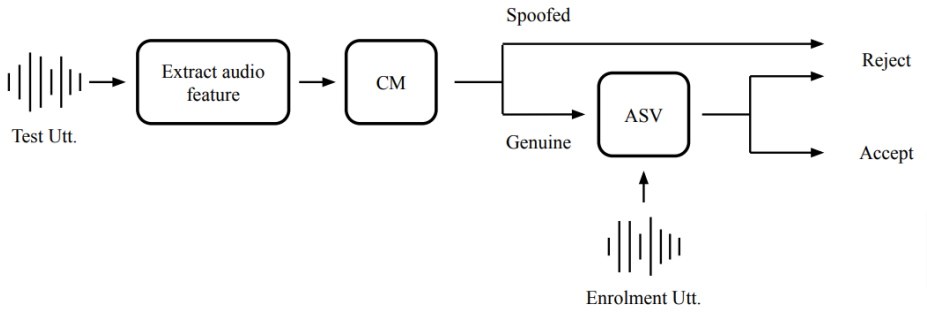


Figure 1.2: Illustration of spoofing countermeasure working in parallel with an ASV system.

function of these CMs is to distinguish between bona fide and spoofed signals, making them speaker- and, optionally, text-independent. Also similar to ASV, CMs produce a score which indicates support for one of two hypotheses, namely that the utterance is bona fide or spoofed. By convention, higher scores indicate greater support for the bona fide hypothesis, while lower scores indicate greater support for the spoof hypothesis.

CMs typically operate in parallel with an ASV system [16], as shown in Figure 1.2, but can also be integrated at the score level [17]. In the more common parallel approach, the CM operates in tandem with the ASV system. If the CM flags an utterance as spoofed, then the tandem system rejects the claimed identity, hopefully resulting a reduced rate of false alarms. Correspondingly, the ASV system only process utterances flagged by CM as genuine (bonafide), it compare the similarity of this genuine utterance and the previously saved enrolment utterance. The tandem system grants access only if the similarity score is high. Conversely, if the similarity score is low, the claimed identity is also rejected.

In the case of score fusion approach, scores of CM and ASV are generated simultaneously and combined to form the final score for the decision making process. The fusion approach differs from the parallel approach that both CM and ASV system contribute to the final decision, both reject or accept, whereas in the parallel approach, both ASV and CM can reject the claimed identity, because either the utterance is spoofed (CM decision) or of different speaker (ASV decision).

In the score fusion approach, CM and ASV systems generate their respective scores simultaneously, which are then combined to establish a final score for the

decision-making process. This fusion strategy stands in contrast to the parallel approach by enabling both the CM and ASV systems to jointly influence the final decision. This differs from the parallel approach, where the CM and ASV systems may independently reject the claimed identity, either due to the CM identifying the utterance as spoofed or the ASV determining a discrepancy in speaker identity.

1.3 Motivation and aim

The field of anti-spoofing has evolved significantly, transitioning from methodologies focusing on hand-crafted features combined with Hidden Markov Models (HMM) [18] and Gaussian Mixture Models (GMM) [17], towards those employing Deep Neural Networks (DNN) [19, 20] such as Convolutional Neural Networks (CNN) [21–23]. However, designing custom network architectures for various hand-crafted features is a time-consuming process [24,25]. Furthermore, the performance of state-of-the-art network architectures can differ based on input features [26]. A potential solution to this issue is to automate the process, allowing the algorithm itself to identify the optimal network architectures for given input features and, going one step further, to determine the optimal combination of architectures and features jointly.

While the discovery of optimal architectures and feature representations may contribute to the design of a better CM system, it does not clarify the underlying rationale [27, 28]. Key questions remain, such as why different features capture different cues [22, 26], or which intervals of the speech signal have the greatest influence upon the score (and decision) [27, 29]. An understanding of these aspects will facilitate the design of more reliable anti-spoofing systems in the future.

Recognizing and understanding the artefacts of spoofing attacks is crucial, yet we've noted that deep-learning-based CMs often display inconsistencies in behavior and performance. This raises questions about whether similar inconsistencies exist within deep-learning-based spoofing attack algorithms and their impact on the robustness of spoofing detection. Recognizing the vulnerability of CMs to these inconsistencies, we propose a strategy to enhance their defensive capabilities. This strategy includes combining and jointly optimising ASV systems and CMs, to improve their ability to detect spoofing attacks. We also analyse our experimental results to further understand the function and behavior of ASV and

CMs in spoofing-aware speaker verification (SASV).

1.4 Contributions

- Introduce evolutive network architecture method, namely Differentiable Architecture Search (DARTS), to automate the architecture learning process for for speech deepfake and spoofing detection.
- Present the first successful application of Partially-Connected DARTS (PC-DARTS) approach to deepfake and spoofing detection problems. The resulting models are not only competitive in performance, but also have reduced computational complexity and GPU memory demands, and require minimal human effort.
- Report the first end-to-end (E2E) solution, namely Raw PC-DARTS, to optimise the network architecture in tandem with front-end feature extraction and network parameters. The performance of this model ranks among the top single-system results reported at the time of publication.
- Introduce the use of SHapley Additive exPlanations (SHAP), a feature attribution method, to highlight the importance of moving beyond black-box models in spoofing detection and emphasise the need for the development of trustworthy, explainable and more robust and reliable spoofing detection systems.
- Demonstrate the first application of SHAP to understand the influences on the outputs of spoofing detection models. Showcase its potential in revealing attention patterns at low-level spectro-temporal intervals.
- Extend the understanding of classifier behavior by employing SHAP for attack analysis, and present findings on the consistencies and differences in artefacts across various spoofing attacks.
- Present first work to demonstrate that the effectiveness of spoofing attacks can vary significantly based on training conditions.
- Introduce the use of jointly-optimised solutions for the design of a robust spoofing detection and speaker verification system. Demonstrate through

experiments that while joint optimisation enhances robustness against spoofing, it tends to compromise speaker verification performance.

- Highlight the need for a joint optimisation approach that capitalises on the complementary information provided by both spoofing detection and speaker verification sub-systems.

- Stress the importance of using auxiliary data from a diverse set of speakers for successful joint optimisation.Highlight the persistent challenge of domain robustness and emphasise the need for research in reducing CM over-fitting and the exploration of speaker-dependent spoofing detection.

1.5 Thesis structure

Presented in this section is the structure of the thesis and content of each chapter. We also cite related, peer-reviewed published work from which the material is drawn.

Chapter 2

This chapter provides background and a literature review. It starts with an overview of the development of ASV systems. Then we present an overview of spoofing attacks and countermeasures. We present different front-end features as well as popular network architectures. We then present details of the corpus and the evaluation metrics used for all experimental work presented in this thesis.

Chapter 3

This chapter presents the first successful application of Differentiable Architecture Search (DARTS) in the study of deepfake and spoofing detection tasks. The goal is to automate and optimise the neural network architecture design, thus minimising human intervention and broadening the use of automatic optimisation beyond model parameters. We conducted a comprehensive investigation, employing a range of methodologies to examine the performance and feasibility of partially-connected DARTS (PC-DARTS) [30] in the realm of deepfake and

spoofing detection. Our findings show promising results, with PC-DARTS demonstrating robust performance compared to the top-performing systems reported in the literature. Performance results from the ASVspoof 2019 logical access (LA) database. Through this study, we show the potential of automatically designed architectures producing neural networks which can compete with, or even outperform their counterparts in the domain of deepfake and spoofing detection.

The work presented in this chapter was published in:

Wanying Ge, Michele Panariello, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Partially-Connected Differentiable Architecture Search for Deepfake and Spoofing Detection**," in *Proc. INTERSPEECH 2021.* Brno, Czech Republic, September 2021.

Chapter 4

The work presented in Chapter 4 builds on the exploration of automatically designed neural architectures from the previous chapter. The central goal of this work is to evolve beyond hand-crafted network models and the front-ends, taking advantage of automation to construct network structures that are capable of processing raw signals. We present our exploration of Raw PC-DARTS. This technique operates directly on raw waveform inputs, allowing for joint optimisation of both network architecture and network parameters. Our findings reveal that, although optimal performance was achieved using a fixed front-end, rather than a learnable configuration, the proposed Raw PC-DARTS system still delivers among the best performance reported at the time of publication for the ASVspoof 2019 LA database.

The work presented in this chapter was published in:

Wanying Ge, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Raw Differentiable Architecture Search for Speech Deepfake and Spoofing Detection**," in *The ASVspoof 2021 Workshop (INTERSPEECH Satellite Workshop).* September 2021.

Chapter 5

This chapter presents our exploration of explainable artificial intelligence (xAI) within the field of deepfake and spoofing detection. Our goal is to decode the "black-box" solutions, which dominate current approaches, and to reveal the features these systems utilise to distinguish between genuine and spoofed speech. We report our use of SHAP, an xAI tool which helps explain the outputs of a detection model. We probe into its potential to uncover why some attacks are more challenging to detect than others. In particular, we are interested to know why some solutions focus more on non-speech intervals. Our findings show that SHAP analysis can highlight the attention a classifier pays to low-level spectro-temporal intervals, demonstrating unexpected classifier behaviours on relying discriminative information from non-speech intervals. We also aimed to recognise a variety of artefacts and assess their significance. Our findings indicated that, despite unique artefacts for each attack, there were shared characteristics among certain attack types and techniques.

The work presented in this chapter was published in:

Wanying Ge, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Explaining Deep Learning Models for Spoofing and Deepfake Detection with SHapley Additive exPlanations**," in *Proc. ICASSP 2022*. Singapore, May 2022.

Wanying Ge, Massimiliano Todisco and Nicholas Evans, "**Explainable Deepfake and Spoofing Detection: An Attack Analysis Using SHapley Additive exPlanations**," in *The Speaker and Language Recognition Workshop*. Beijing, China, June 2022.

Chapter 6

In this chapter, we explore the vulnerability of CMs to spoofing attacks generated by deep learning models with subtle variations in their training configurations. Although we have successfully identify attack-specific artefacts in the previous

chapter, these artefacts may not be the sole identity of the algorithm since both CM and attacks are now deep-learning-dominated and we already observed in the previous chapters that most CMs suffer from inconsistency in model behaviour and performance. The findings and observations presented in this chapter have propelled our investigation into joint optimisation in the next chapter.

The work presented in this chapter was published in:

Wanying Ge, Xin Wang, Junichi Yamagishi, Massimiliano Todisco and Nicholas Evans, "**Spoofing Attack Augmentation: Can Differently-trained Attack Models Improve Generalisation?**," in *Proc. ICASSP 2024*, Seoul, South Korea, April 2024.

Chapter 7

This chapter presents our investigation into the Spoofing-Aware Speaker Verification (SASV) systems. The motivation behind this study is to understand why, despite that ASV and CM working in synergy has the potential of utilising complementarity information from both systems, the joint optimisation of speaker verification and spoofing detection subsystems was unsuccessful in the first SASV challenges. Our findings reveal that, while joint optimisation indeed improves robustness to spoofing, it also degrades speaker verification performance. This trade-off suggests that an effective joint optimisation strategy must consider the complementary information provided by each subsystem. The work in this chapter also highlights the issue of overfitting to speaker data. We demonstrate that with sufficient speaker data, joint optimisation can yield superior SASV performance, even outperforming separately optimised systems. Thus, despite potential drawbacks, the evidence points towards the advantages of joint optimisation in developing more reliable speaker verification systems.

The work presented in this chapter was published in:

Wanying Ge*, Hemlata Tak*, Massimiliano Todisco and Nicholas Evans, "**On The Potential of Jointly-Optimised Solutions to Spoofing Attack Detection and Automatic Speaker Verification**," in *Proc. IberSPEECH*

2022. Granada, Spain, November 2022 (* equal contribution).

Wanying Ge, Hemlata Tak, Massimiliano Todisco and Nicholas Evans, "**Can Spoofing Countermeasure and Speaker Verification Systems Be Jointly Optimised?**" in *Proc. ICASSP 2023*. Rhodes Island, Greece, June 2023.

Publications

1. **Wanying Ge**, Michele Panariello, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Partially-Connected Differentiable Architecture Search for Deepfake and Spoofing Detection**," in *Proc. INTERSPEECH 2021*, Brno, Czech Republic, September 2021.

2. **Wanying Ge**, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Raw Differentiable Architecture Search for Speech Deepfake and Spoofing Detection**," in *The ASVspoof 2021 Workshop (INTERSPEECH Satellite Workshop)*, September 2021.

3. **Wanying Ge**, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Explaining Deep Learning Models for Spoofing and Deepfake Detection with SHapley Additive exPlanations**," in *Proc. ICASSP 2022*, Singapore, May 2022.

4. **Wanying Ge**, Massimiliano Todisco and Nicholas Evans, "**Explainable Deepfake and Spoofing Detection: An Attack Analysis Using SHapley Additive exPlanations**," in *The Speaker and Language Recognition Workshop*, Beijing, China, June 2022.

5. **Wanying Ge***, Hemlata Tak*, Massimiliano Todisco and Nicholas Evans, "**On The Potential of Jointly-Optimised Solutions to Spoofing Attack Detection and Automatic Speaker Verification**," in *Proc. IberSPEECH 2022*, Granada, Spain, November 2022, Best Paper Award (* equal contribution).

6. **Wanying Ge**, Hemlata Tak, Massimiliano Todisco and Nicholas Evans, "**Can Spoofing Countermeasure and Speaker Verification Systems**

Be Jointly Optimised?," in *Proc. ICASSP 2023*, Rhodes Island, Greece, June 2023.

7. **Wanying Ge**, Xin Wang, Junichi Yamagishi, Massimiliano Todisco and Nicholas Evans, "**Spoofing Attack Augmentation: Can Differently-trained Attack Models Improve Generalisation?**," in *ICASSP 2024*, Seoul, South Korea, April 2024.

Other work

1. Michele Panariello*, **Wanying Ge***, Hemlata Tak, Massimiliano Todisco and Nicholas Evans, "**Malafide: A Novel Adversarial Convolutive Noise Attack Against Deepfake and Spoofing Detection Systems**," in Proc. *INTERSPEECH 2023*, Dublin, Ireland, August 2023 (*equal contribution).

In above manuscript, I was involved in discussion, experiment and paper writing.

Chapter 2

Literature review

This chapter delves into the literature review of specific components integral to deep-learning-based ASV and spoofing CM solutions, which are central to the remaining chapters of this thesis. Section 2.1 introduces the database collection for both ASV and CM, highlighting the similarities and differences in data for training and evaluation due to the distinct nature of the tasks. The integration of these tasks and the inspirations for Chapter 6 and 7 are also covered. Section 2.2 discusses the feature extraction essential for representing the audio elements of the data, along with tools highlighted in Chapter 5 to illustrate the significance of these audio elements. Lastly, Section 2.3 focuses on model design, presenting works on creating deep learning models adept at extracting relevant information from audio features, as well as the automated design and search for such tools, forming the basis for Chapter 3 and 4.

2.1 Database characteristics and their impact

We first present an overview of the databases used in this work, namely the VoxCeleb1 and VoxCeleb2 for speaker verification and the ASVspoof 2019 Logical Access and the Fake Audio Detection databases for spoofing detection. We also describe other related data augmentation techniques and non-speech databases for improving system robustness.

2.1.1 Databases for ASV and CM: An overview

The ASV system primarily functions as a biometric security tool, verifying speaker authenticity in contexts like voice dialling, voicemail, security control, online bank-

Table 2.1: Comparative overview of VoxCeleb1 and VoxCeleb2 databases: languages, speaker, utterances, and total hours.

	VoxCeleb1	**VoxCeleb2**
Language	Mostly English	
# of Spks	1,251	6,112
# of Utters	153,516	1,128,246
# of Hours	352	2,442

ing, telephone shopping, and forensic applications by extracting unique speaker representations from speech signals [31–34]. We showcase statistics from two speaker recognition databases, VoxCeleb1 [31] and VoxCeleb2 [32], in Table 2.1. These databases utilise a multi-stage data collection process, initially choosing Persons of Interest (POIs) mostly from English speakers. They employ face recognition to confirm detected face identities and active speaker verification to align the visible face with the speech source. As a result, they offer robust and varied speaker recognition datasets. Despite being primarily English-based, the databases in Table 2.1 encompass data from over 7000 POIs spanning 145 nationalities, accounting for diverse accents, ages, and ethnic backgrounds. Such extensive databases are crucial for speaker recognition, as they provide a wide range of training data, fostering the development of more robust and generalised models.

Spoofing countermeasure (CM) systems play a crucial role in identifying and mitigating spoofing attacks. The function of CM is to extract distinctive features and patterns from speech signals to analyse and identify telltale signs of spoofing artefacts [14, 35, 36]. Table 2.2 provides statistics for two databases, ASVspoof 2019 Logical Access (LA) [15] and the Fake Audio Detection (FAD) [35]. Unlike speaker recognition databases, these databases contain both genuine and spoofed audio samples, as they designed to aid in the development of countermeasures against spoofing attacks. While speaker recognition databases typically comprise a vast array of speakers varying in gender and age to discern differences among them [34, 37], spoofing and deepfake detection databases have fewer, meanwhile emphasising diverse spoofing attack types.

Furthermore, Table 2.2 reveals that both databases cover known and unknown

Table 2.2: Comparative Analysis of ASVspoof 2019 LA and FAD databases: language, speaker, and attack types in training and evaluation.

	ASVspoof 2019 LA	**FAD**
Language	English	Chinese
# of Spks	107	1024
# of Atks in Train	4 TTS, 2 VC	8 TTS
# of Atks in Eval	7 TTS, 3 VC, 3 TTS-VC	11 TTS

spoofing attack types. This inclusion allows researchers to assess their systems' efficacy against known threats and their adaptability in detecting new attack types. A system that can successfully detect and counter unknown attacks is significantly more valuable in maintaining security than a system that can only handle known threats.

2.1.2 Towards further improvement on robustness

While increasing data diversity typically results in more robust systems, databases like VoxCeleb and ASVspoof, despite their diversity, have limited data due to the extensive effort required for their collection and maintenance. Consequently, non-speech-based databases and signal processing techniques have been introduced for data augmentation to further enhance system robustness. For ASV system training, techniques and resources like Room Impulse Response (RIR) [38], MUSAN corpus [39], pitch shifting [40], and SpecAugment [41] are employed. In contrast, CM system training commonly utilises signal processing techniques [40–42] and artificial manipulation of the provided data [43, 44]. They all serve as a means to create a more diverse and challenging set of training examples and help improve the robustness of the systems.

Training data for ASV systems is designed to capture the variability among real speakers, the resulting ASV system is thus vulnerable to high-quality synthesised or replayed speeches that imitate voices of a real, target speaker [15]. To protect ASV from such threat, CM training data is specifically collected to aid systems on distinguishing genuine speeches from deepfakes. While the CM training data already covers a wide range of attacks, we argue that this approach may not be enough for ensuring generalisation and robustness. In Chapter 6, we discuss

the observation that the efficacy of deep-learning-based CM solutions varies significantly based on factors like initialisation [26], hyper-parameters [45], and data partitions [43]. Consequently, the potency of spoofing attacks, particularly those using deep learning, will also differ depending on their training conditions. This inconsistency might lead CM to occasionally overlook certain attack algorithms, including those it has been trained on.

Understanding CM's potential vulnerabilities suggests that ASV systems could be trained or refined to defend specifically against threats that bypass CM, acting as additional protection. While ASV is vulnerable to some spoofing attacks, it still can sometimes detect fake speeches that poorly imitate the original speaker's characteristic. This mutual support implies that both systems can be jointly optimised for heightened security. Events such as the Spoofing-Aware Speaker Verification challenge [46] were initiated to probe this concept. Nevertheless, it's notable that many leading solutions in the challenge [47–51] primarily employ ASV and CM systems that are trained separately and later integrated for combined functionality. We thus delve into the possibility of optimising CM and ASV systems synergistically in Chapter 7 for the designing of a more robust integrated systems.

2.2 Feature extraction and importance

This section delves into the important role of feature extraction in enhancing the performance of ASV and CM systems, focusing on the methodologies employed for effective speaker recognition and anti-spoofing. We first introduce the fundamental processes and methodologies used in feature representation for ASV and CM. These methodologies underscore the importance of extracting compact, yet comprehensive, information from speech signals to distinguish between speakers accurately. Following this, we explore the significance of understanding and evaluating feature importance within these systems. This exploration not only aids in refining system performance but also contributes to the broader goal of advancing explainable AI within the field.

2.2.1 Methodologies in feature representation for ASV and CM

Most feature extraction stages of current speaker recognition and verification systems function in a similar way - they are designed to extract representative and compact information for the speech signals for the discrimination of different speakers. The traditional Gaussian Mixture Model - Universal Background Model (GMM-UBM) is a foundational method in speaker recognition [4]. It posits that individual speaker characteristics can be represented by various Gaussian distributions. The system encompasses the Universal Background Model (UBM), a generalised model that captures the collective acoustic attributes of speakers, and individualised Gaussian Mixture Models (GMMs). These GMMs, tailored from the UBM emphasise the unique vocal features of each speaker. Recently, deep learning has introduced significant advancements in speaker recognition. The i-vector framework [52] offer enhanced modelling of intricate data distributions, capturing detailed phonetic specifics, leading to notable improvements in recognition accuracy, especially in challenging environments [53]. The DNN-BNF/i-vector model further refines this by extracting compact features from a DNN's bottleneck layer [54, 55], resulting in a superior performance over traditional methods.

In anti-spoofing, accurately representing voice signals is pivotal for detecting spoofs. The process involves transforming raw voice signals into features that capture essential characteristics to differentiate genuine from spoofed speech. Commonly used features include the Short-Time Fourier Transform (STFT) magnitude [22, 26], which shifts the speech signal from time to frequency domain, highlighting crucial elements like formants and harmonics. Another method, Linear Frequency Cepstral Coefficients (LFCC) [56, 57], is akin to Mel Frequency Cepstrum Coefficient (MFCC) but focuses on a linear frequency scale, capturing high-frequency components more effectively. Constant-Q Cepstral Coefficients (CQCC) [24], on the other hand, offers both temporal and spectral characteristics, proving invaluable in anti-spoofing. The emerging trend of end-to-end (E2E) solutions combines feature extraction and model training into a unified process [58,59]. This approach allows models to derive optimal features from raw data, potentially improving performance [21, 60, 61].

2.2.2 Understanding and evaluating feature importance

Despite many years of research into voice anti-spoofing, we still don't fully understand the specific clues that systems use to differentiate real from fake voices. It's essential to comprehend how these classifiers make decisions to trust them and further the move towards explainable AI. Feature attribution method offer insights into DNN-based detection by highlighting the significance of individual input features in these decisions. Understanding these cues can improve system performance and is crucial for scenarios like forensics.

Several prior studies have explored explainability in related speech topics. For instance, [62] uses an approach based on attenuating specific spectral components, demonstrating that artefacts indicative of different spoofing attacks are found within distinct sub-band intervals and can be detected more reliably by front-ends that emphasise the corresponding frequency range. Grad-CAM (Gradient-weighted Class Activation Mapping) [63] has been employed to explain spoofing classifier behaviour in [64], generating a binary saliency map for the network input layer. By reconstructing input audio using spectrograms masked with the binary saliency map, listening experiments reveal that the model distinguishes between genuine and spoofed speech based on buzziness and rhythmic quality. A study of replay detection [29] examines the impact of various replay attack configurations on detection performance, while another study [65] utilises LIME (Local Interpretable Model-agnostic Explanations) [66] to generate temporal and spectral explanations for voice replay detection model prediction behaviour. These works suggest that non-speech intervals can provide discriminative information for spoofing detection. Further research [27] indicates that the duration of non-speech intervals in a synthetic speech and converted voice detection task can also hint at whether an utterance is genuine or spoofed.

Inspired by cooperative game theory concepts [67], SHapley Additive exPlanations (SHAP) [68] offers a more sophisticated and potent approach to explainability. SHAP values indicate the contribution of specific features to a classifier output. The study in [69] employs DeepSHAP [68] to elucidate the behaviour of speech enhancement models, using SHAP values to identify regions of the input feature (in the form of spectrograms) that most significantly impact the model output. The findings reveal that higher-performing models tend to rely more on

information within speech-dominated spectro-temporal intervals. As a unified and theoretically grounded method, SHAP can explain the relative importance of particular features to classifier outputs.

2.3 Model architecture

2.3.1 Evolution of model architectures for ASV and CM

Deep neural networks (DNNs) in speaker recognition use various structures such as DNN, CNN, and Long short-term memory (LSTM), each with unique components and methods to capture speaker information. These structures, combined with a range of input options, enhance recognition performance. While CNNs frequently employ acoustic data like mel-filterbanks [70, 71], other models extract features directly from raw waveforms [59, 72]. The selection of input, influenced by the task and network type, with each offering different audio representations. Critical to this process are temporal pooling layers [73, 74], which transform frame-level features into utterance-level summaries using techniques like average pooling and self-attention-based pooling, emphasising essential frames for speaker identification. Objective functions, including softmax loss [9] and angular softmax loss [75], guide networks to learn distinguishable speaker embeddings. In sum, the structure and input capture speech data, pooling layers summarise this data, objective functions guide network training. Together, these elements make deep embedding models more effective in speaker recognition tasks.

Voice anti-spoofing technology has evolved considerably over time. Initially, the field relied on CQCC-GMM [24], which prioritized frequency and temporal resolutions for better spoofing detection. Later, ResNet, or Residual Networks [76], introduced skip connections to combat model degradation, enabling the training of deeper networks [57, 77]. Despite advancements achieved with ResNet and its variants [57, 77], some spoofing methods remained difficult. This led to the development of RawNet2 [60], which operated on raw speech waveforms, eliminating the need for hand-crafted features. The most recent E2E model is AASIST [78], optimised for real-world applications, integrating both spectral and temporal data with advanced graph attention layers. Currently, self-supervised learning (SSL) [79, 80] offers state-of-the-art performance in the anti-spoofing domain. Using wav2vec 2.0 front-end with fine-tuning, SSL first trains on large volumes of unlabeled genuine

speech, followed by fine-tuning on labeled datasets containing both genuine and spoofed speech. This method potentially enhances the model's adaptability to unseen spoofing attacks.

2.3.2 Automated model architecture design

End-to-end (E2E) processing techniques have has gained increasing attention [21, 60, 78, 81], and have now replaced the use of hand-crafted, manually optimised components with automated learned representations. While these E2E approaches have indeed set benchmarks in performance, it's noteworthy that their advancements predominantly cater to the front-end components. When it comes to back-end components, optimisation generally narrows down to refining network parameters. The architecture of these networks often remains hand-crafted. Building upon earlier work [82], the use of neuro-evolution for augmenting topologies (NEAT) [82] to learn network architectures automatically reported studied in [83]. However, performance was below the state-of-the-art, and computational complexity was high. Instead of pursuing more efficient NEAT implementations, we turned to powerful and efficient alternatives with proven potential in speech-related tasks.

In Chapter 3 and 4, we have investigated the application of neural architecture search (NAS), first proposed in [84]. NAS methods compose of an architecture search space, a search strategy, and an evaluation strategy [85]. The search space contains a collection of candidate operations such as dilated convolution, separable convolution and average pooling. An architecture is generated from this space based on the performance criteria. The NAS variation known as differentiable architecture search (DARTS) [86] facilitates the selection of candidate operations by using a search space with continuous and learnable weights. DARTS models can be optimised using backpropagation with hardware acceleration. The network is designed automatically by optimising the operations within architecture building blocks called cells. During an initial search phase, candidate operations such as convolutional operations, pooling layers, and residual connections are selected. Following this, the resulting cells are combined to develop a more complex architecture, which is then further optimised in sequential phases.

DARTS has been applied successfully to in various speech and language tasks, as evidenced by its application in multiple studies [87–89]. In one instance, DARTS

was used for architecture search in a keyword spotting task [87]. This study achieved competitive results by employing a search space containing the standard operations used in ResNet. Another successful application of DARTS, in automatic speech recognition, was reported in [88]. This study showed promising results even when the architecture search and training stages were conducted using different language datasets. The first implementation of DARTS in speaker verification is documented in [89]. The findings indicate that smaller, automatically learned solutions are comparable to their hand-crafted counterparts. Both [87] and [89] report state-of-the-art performance results. However, they also highlight the necessity of using small batch sizes to enable architecture search on a single GPU.

2.4 Performance metrics

Two performance metrics are utilised in our experiment to evaluate the effectiveness of speaker verification and spoofing detection systems, namely the Equal Error Rate (EER) and the Tandem Detection Cost Function (t-DCF). The EER is a widely accepted metric for assessing a model's capability to distinguish between target and non-target speaker utterances, or between bona fide and spoofed utterances in spoofing scenarios. On the other hand, the t-DCF provides a comprehensive evaluation of the joint performance of countermeasure mechanisms and automatic speaker verification systems under spoofing conditions.

2.4.1 Equal error rate

EER is a common evaluation metric utilised in both speaker verification and spoofing detection scenarios. It is used to measure the model's ability to differentiate between target speaker and non-target speaker utterances, or in the case of spoofing detection, bona fide and spoofed utterances. The EER essentially represents the error rate at an operating point where both the False Acceptance Rate (FAR) and False Rejection Rate (FRR) are equal. FAR and FRR are calculated as follows:

$$FAR = \frac{FP}{FP + TN} \tag{2.1}$$

$$FRR = \frac{FN}{TP + FN} \tag{2.2}$$

where FP is the count of False Positives, TN is True Negatives, FN is False Negatives and TP is True Positives.

2.4.2 Tandem detection cost function

Unlike the EER, the Tandem Detection Cost Function (t-DCF) [90] is primarily used in spoofing detection scenarios. It evaluates the joint performance of both the countermeasure and automatic speaker verification (ASV) system under a Bayesian decision risk approach. The t-DCF metric is calculated as follows:

$$\begin{aligned} t-DCF(s,t) =& C_{asv_{miss}} \cdot \pi_{tar} \cdot P_a(s,t) \\ &+ C_{asv_{fa}} \cdot \pi_{non} \cdot P_b(s,t) \\ &+ C_{cm_{fa}} \cdot \pi_{spoof} \cdot P_c(s,t) \\ &+ C_{cm_{miss}} \cdot \pi_{tar} \cdot P_d(s) \end{aligned} \tag{2.3}$$

where $C_{asv_{miss}}$ and $C_{asv_{fa}}$ are the costs of the ASV system rejecting a target trial and accepting a non-target trial respectively, $C_{cm_{fa}}$ and $C_{cm_{miss}}$ are the costs of the CM accepting a spoof trial and rejecting a human trial respectively, π_{tar}, π_{non}, π_{spoof} are the prior probabilities of target, non-target, and spoof respectively. $P_a(s,t)$, $P_b(s,t)$, $P_c(s,t)$ and $P_d(s)$ represent different error probabilities which are a function of detection thresholds for a CM and an ASV system respectively. A reference value of 1.00 for (normalised) t-DCF indicates an uninformative countermeasure. Hence, the lower the t-DCF value, the better the countermeasure's performance.

2.5 Summary

This chapter provides an overview of the common databases for automatic speaker verification and spoofing detection. The work described in Chapter 3 to Chapter 5 uses the ASVspoof 2019 LA database and the work described in Chapter 7 uses VoxCeleb1&2, ASVspoof 2019 LA and the FAD database. This chapter also presents an literature review on feature extraction and network design of both ASV and CM systems. Finally, this chapter provides an introduction of two commonly used performance metrics that will be used in this thesis to evaluate countermeasure performance.

Chapter 3

Neural architecture search for spoofing detection

In this chapter, we introduce the first successful implementation of a differentiable architecture search (DARTS) [86] method to address deepfake and spoofing detection tasks. Section 3.1 presents our motivation of applying neural architecture search methods in the field of spoofing detection. Section 3.2 gives a theoretical introduction to the DARTS algorithm. Section 3.3 describes the experimental setup and details. Section 3.4 reports our results. Section 3.5 presents the summary of the work.

The work presented in this chapter was published in:

Wanying Ge, Michele Panariello, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Partially-Connected Differentiable Architecture Search for Deepfake and Spoofing Detection**," in *Proc. INTERSPEECH 2021*.

3.1 Introduction and motivation

DARTS, a type of neural architecture search algorithm, functions within a continuous and differentiable search space, facilitating the joint optimisation of network architecture and parameters using gradient descent. Inspired by the successful studies of DARTS in speech related topics [87–89], our work presented in this chapter aims to achieve several objectives. First, we seek to determine whether neural architectures learned automatically with PC-DARTS can outperform hand-crafted networks. Second, we aim to explore the long-term potential for such networks to surpass the current state-of-the-art. Third, we are interested in determining

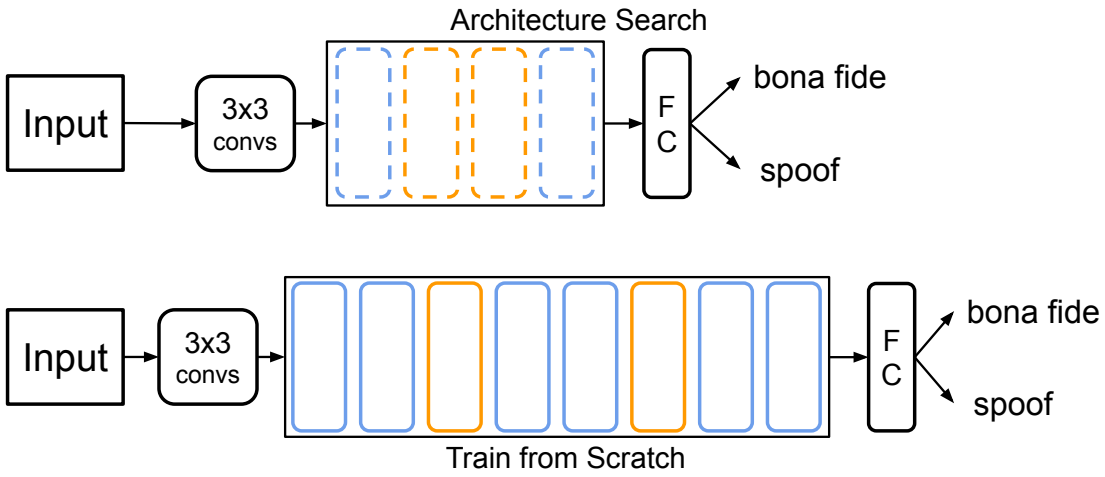


Figure 3.1: An illustration of architecture search and train from scratch. Architecture search optimises a stack of 2 normal cells (dashed blue) and reduction cells (dashed yellow). The train from scratch stage optimises a deeper network of stacked cells (solid blue and yellow). Only network parameters are optimised in the second stage; the cell architectures are those fixed during architecture search.

whether automatically learned and optimised solutions offer greater efficiency. Although not a primary goal of our work, we hypothesise that PC-DARTS might produce less complex networks that exhibit more easily explainable behaviour.

By using random channel masking in the search domain, partially-connected DARTS (PC-DARTS) [30] decreases GPU consumption and autonomously discovers and enhances complex neural architectures composed of convolutional processes and residual blocks. These rapidly learned networks, require minimal human input and demonstrate competitive performance when compared to the top-performing systems documented in the literature at the time of publication, as evidenced by the ASVspoof 2019 LA [15] evaluation set.

3.2 Automatic search of network architectures

Figure 3.1 demonstrates that DARTS consists of two distinct learning stages: the *architecture search* stage (shown to the top) and the *train from scratch* stage (shown to the bottom). A key principle is to develop an intricate network architecture by employing two *cells* (blue and yellow blocks in Figure 3.1). The structure and parameters of these cells are learned automatically.

DARTS distinguishes itself from other NAS techniques by working within a flexible, continuous search space rather than searching over a discrete collection of candidate network operations. This flexibility makes the architecture representation *differentiable*, allowing it to be optimised using conventional methods such as gradient descent, backpropagation and hardware acceleration. During the architecture search stage, the *architecture parameters* of the cells are learned and then fixed in the next stage. The train from scratch stage involves constructing a deeper network by stacking a larger number of cells obtained from the previous stage. *Network parameters* are then re-optimised.

The initial architecture search phase demands significant computational resources. Employing partial connections [30] offers a more resource-efficient solution. These two algorithms are introduced in Section 3.2.1 and Section 3.2.2 respectively.

3.2.1 Differentiable architecture search

Figure 3.2 shows an example of how DARTS networks are formed through the concatenation of multiple *cells*, the internal architectures of which are learned automatically. These architectures determine the use of candidate operations which are applied to input data to generate the output.

Each cell is comprised of N nodes, with each node $\mathbf{x}^{(i)}$ representing a feature map in tensor form. The initial node pair, $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, serve as the cell inputs and connect to the outputs of the two previous cells. Intermediate nodes, ranging from $\mathbf{x}^{(3)}$ to $\mathbf{x}^{(N-1)}$, are computed from prior nodes using operation o chosen from the search space $\mathcal{O}$ as follows:

$$\mathbf{x}^{(j)} = \sum_{i<j} o^{(i,j)} \left(\mathbf{x}^{(i)}\right) \tag{3.1}$$

Here, $o^{(i,j)}$ denotes the operation performed on edge (i, j), connecting $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. The final node, $\mathbf{x}^{(N)}$, represents the cell output, and its feature map is derived from the concatenation of feature maps associated with all intermediate nodes.

During the architecture search stage, a linear combination of operations, denoted as $\bar{o}$, is executed on edge (i, j) based on a weight $\alpha_o^{(i,j)}$. These weights

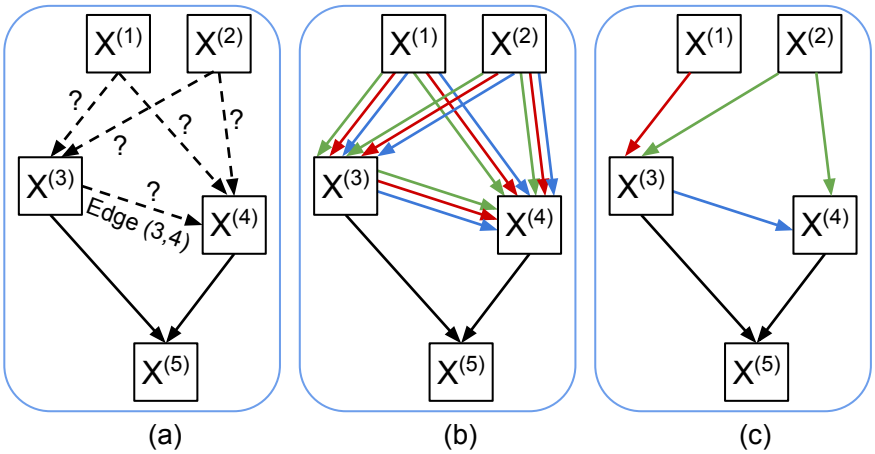


Figure 3.2: An illustration of architecture search: (a) a neural cell with $N = 5$ nodes; (b) an illustration of the candidate operations performed on each edge that are optimised during architecture search; (c) resulting optimised cell with $K = 2$ inputs to each intermediate node.

generate a continuous search space via a softmax function:

$$\bar{o}^{(i,j)}\left(\mathbf{x}^{(i)}\right) = \sum_{o \in \mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)}, o\left(\mathbf{x}^{(i)}\right) \tag{3.2}$$

Consequently, architecture search involves learning a set of continuous variables $\boldsymbol{\alpha} = \{\alpha^{(i,j)}\}$, where $\alpha^{(i,j)}$ has a dimension of $|\mathcal{O}|$. Both the *architecture parameters* $\boldsymbol{\alpha}$ and the *network parameters* $\boldsymbol{\omega}$ (e.g., convolutional filter weights) can be optimised jointly through backpropagation. The objective is to optimise $\boldsymbol{\alpha}$ that minimise the validation loss L_{val}, while also optimising $\boldsymbol{\omega}$ to minimise the training loss:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} L_{val}(\boldsymbol{\omega}^*, \boldsymbol{\alpha}) \\ &\text{s.t. } \boldsymbol{\omega}^* = \operatorname*{argmin}_{\boldsymbol{\omega}} L_{train}(\boldsymbol{\omega}, \boldsymbol{\alpha}) \end{aligned} \tag{3.3}$$

When the search phase is completed, $\bar{o}^{(i,j)}$ is substituted by the single operation with the highest value $\alpha_o^{(i,j)}$. The final cell architecture is derived by preserving the K edges with the highest weights $\alpha_o^{(i,j)}$ that enter each intermediate node, where K is a hyperparameter. All other edges are discarded.

The search space $\mathcal{O}$, as suggested in [89], includes the following set of operations: 3×3 separable convolution; 5×5 separable convolution; 3×3 dilated

convolution; 5×5 dilated convolution; skip connection; 3×3 average pooling; 3×3 max pooling; none (no connection). These operations can form two categories of neural cells: *normal* cells and *reduction* cells. As shown at the bottom of Figure 3.1, cells are stacked to create the complete, deeper residual network. Reduction cells are positioned at $\frac{1}{3}$ and $\frac{2}{3}$ of the overall network depth (total number of stacked cells). The feature map dimensions remain constant for the input and output of each normal cell, while reduction cells reduce the feature map dimensions by 50% and simultaneously double the number of channels.

3.2.2 Partial channel connections and edge normalisation

The computational demand of DARTS, particularly during the architecture search stage, remains high. To address the efficiency issue, we employ partial channel connections and edge normalisation, as proposed in [30]. Partially-connected DARTS (PC-DARTS) offers considerable reductions in computation and memory usage. For any given (i, j), partial channel connections are established through the element-wise multiplication of $\mathbf{x}^{(i)}$ by a masking operator $\mathbf{S}^{(i,j)}$ with the same dimensions. This masking operator either *selects* (multiplying by 1) or *masks* (multiplying by 0) each channel in $\mathbf{x}^{(i)}$:

$$\bar{o}^{(i,j)}\left(\mathbf{x}^{(i)}\right) = \sum_{o \in \mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} o\left(\mathbf{S}^{(i,j)} \odot \mathbf{x}^{(i)}\right) + \left(1 - \mathbf{S}^{(i,j)}\right) \odot \mathbf{x}^{(i)} \tag{3.4}$$

where $\odot$ represents element-wise multiplication. A hyperparameter K_C is defined to preserve a random proportion $1/K_C$ of the available channels. Consequently, partial connections decrease the computational burden by a factor of K_C while simultaneously regularising the selection of weight-free candidate operations (such as max pooling) in $\mathcal{O}$ for a specific edge [30]. A balance should be made between performance (smaller K_C) and efficiency (larger K_C). Due to random channel sampling, the linear combination of operations $\bar{o}^{(i,j)}$ for each node may become unstable during training. To address this problem, a set of *edge normalisation* parameters $\boldsymbol{\beta}$ is introduced to stabilise node inputs as follows:

$$\mathbf{x}^{(j)} = \sum_{i<j} \frac{\exp\left(\beta^{(i,j)}\right)}{\sum_{i'<j} \exp\left(\beta^{(i',j)}\right)} \bar{o}^{(i,j)}\left(\mathbf{x}^{(i)}\right) \tag{3.5}$$

where $\beta^{(i,j)}$ is a learnable smoothing factor. By minimizing L_{val}, the collection of architecture parameters being optimised now includes both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

3.3 Experimental setup

Section 3.3.1 presents a description of the database, protocols and metrics. Input features are described in Section 3.3.2. Details of model training are described in Section 3.3.3.

3.3.1 Database, protocols and metrics

All experiments presented in this chapter were performed using the ASVspoof 2019 Logical Access (LA) database [91]. During the architecture search phase, 50% of the utterances for each class in the training partition, i.e. both bona fide and spoofed instances (A01-A06), are employed for learning network parameters. The remaining 50% serve to learn architectures, specifically one normal cell and one reduction cell. Subsequently, the cell architectures yielding the highest classification accuracy are used in the training from scratch stage.

After the training from scratch phase, the resulting model's performance is evaluated using the complete evaluation partition. Performance metrics include the pooled minimum normalised tandem detection cost function (min-tDCF) [92] as well as the pooled equal error rate (EER).

3.3.2 Input feature

Our initial experiments showed that applying neural architecture search to raw audio waveforms requires significant GPU memory, resulting in reduced batch sizes and extended training time [60]. Consequently, we employed 60-dimensional linear frequency cepstral coefficients (LFCCs), including static, delta, and delta-delta coefficients. Feature extraction utilised 64 ms Hamming windows with a 16 ms shift and a 1024-point FFT. To encourage generalisation, frequency masking was implemented following the method outlined in [77], with up to 12 masked frequency channels per mini-batch.

3.3.3 Model training

Following standard practice [30], we applied three convolutional layers with a stride of 2 to input features for resolution reduction. Architecture search utilised 4 neural cells (2 normal cells and 2 reduction cells) with 16 initial channels. For each intermediate node, each cell contains $N = 7$ nodes and retaining $K = 2$ inputs after the architecture search stage.

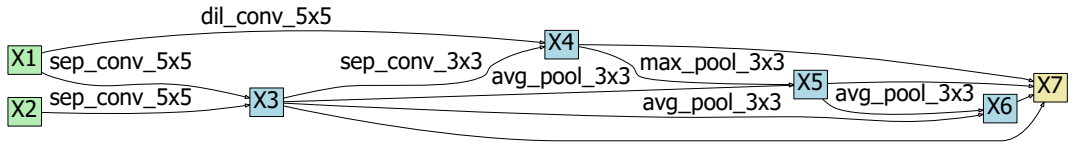
The architecture search stage involved 50 epochs of training with a 64 batch size. Adam optimiser [93] is used to learn both architecture and network parameters. These parameters were optimised by minimising the weighted cross-entropy loss between spoofed and bona fide data at a 1:9 ratio. Following [30, 94], architecture parameters were not updated during the initial 10 epochs. Architecture parameter learning used a 6e-4 learning rate and 0.001 weight decay, while network parameter learning used a 0.01 initial learning rate, which was annealed to 0.001 following a cosine schedule. Partial channel connections applied a $K_C = 2$ value. Upon completion of the architecture search stage, network parameters $\boldsymbol{\omega}$ were discarded, retaining only normal and reduction cell architectures.

3.4 Results

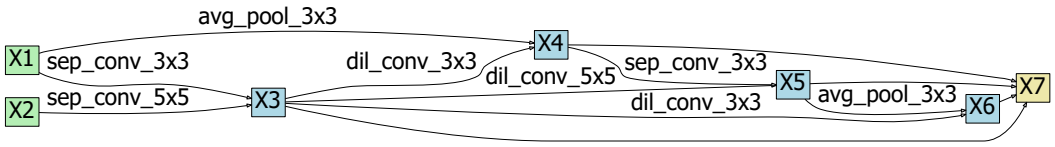
We present our experimental results in this section. Sec. 3.4.1 shows results obtained in the architecture search stage, as well as the found network architectures. Sec. 3.4.2 reports results obtained in the train from scratch stage. Performance comparison to other systems are reported in Sec. 3.4.3.

3.4.1 The searched architecture

The architecture search stage has the highest computational demand. Consequently, we focus on both search time and performance, both are shown in Table 3.1 for experiments involving DARTS and PC-DARTS with models featuring 4 layers and 16 channels ($L = 4, C = 16$). For DARTS, the batch size is maximised based on GPU memory limitations. Implementing partial connections reduces search time by approximately 50%, while regularisation enhances accuracy. Performance effectively translates from the training partition to the development partition. Resulting normal and reduction cell architectures are shown in Figure 3.3.



(a) Normal cell



(b) Reduction cell

Figure 3.3: An illustration of the (a) normal and (b) reduction cells resulting from architecture search. As illustrated in Figure 3.1, they form the basic building blocks used to construct the architecture used in the train from scratch stage.

Table 3.1: A comparison of DARTS and PC-DARTS models with $L = 4$ layers and $C = 16$ channels. Results in terms of processing efficiency (GPU-days) and accuracy for ASVspoof 2019 LA training and development partitions.

Model size	**Systems**	Search Cost	Best Architecture	
		GPU-days	**Train Acc.**	**Dev Acc.**
($L = 4$,	DARTS	0.29	98.80%	97.21%
$C = 16$)	PC-DARTS	0.15	99.97%	100%

3.4.2 Train from scratch

The resulting cell architectures in Section 3.4.1 can be used in the second stage to form a new deep network with arbitrary numbers of layers (L) and initial channels (C). Table 3.2 presents results for various PC-DARTS configurations (column 1) and number of parameters (column 2), with min-tDCF and EER results for both the development partition (columns 3 and 4) and evaluation partition (columns 5 and 6). The top-performing model, as per the primary min-tDCF metric in evaluation set, has 16 layers and 64 initial channels, yielding a min-tDCF of 0.0914 and an EER of 4.96 for the evaluation partition. The second-best model, which has 4 layers and 16 initial channels, achieves a min-tDCF of 0.0992 and an EER

Table 3.2: Number of parameters and results for a selection of different PC-DARTS models. Results for the ASVspoof 2019 LA database.

Model size	**Params**	Dev		Eval	
		min-tDCF	**EER**	**min-tDCF**	**EER**
$(L = 2, C = 4)$	0.007M	0.0004	0.04%	0.1244	5.80
$(L = 4, C = 16)$	0.14M	0	0	0.0992	5.53
$(L = 8, C = 32)$	0.97M	0.00004	0.002	0.1177	4.87
$(L = 16, C = 64)$	7.51M	0	0	0.0914	4.96
$(L = 24, C = 64)$	10.57M	0.0001	0.039	0.1045	5.51

of 5.53 with 7.37M fewer parameters. While the performance of the smallest model is considerably worse in terms of min-tDCF, its EER remains respectable. The largest tested model size provides no performance improvement, likely due to overfitting to the training data.

3.4.3 Comparison to competing systems

Table 3.3 shows results with top-performing systems from the literature at the time and the two ASVspoof 2019 baselines [15]. The optimal (16,64) model substantially outperforms the two ASVspoof baselines and all but two other models, both of which are Res2Net models [95]. Although differences in min-tDCF are modest, they are more pronounced in terms of EER. Our second-best model, utilising 85% fewer parameters than the best-performing Res2Net model, remains competitive. These encouraging results demonstrate that anti-spoofing models, with both *architecture* and *parameters* learned automatically, can outperform models designed with extensive human effort.

3.5 Conclusion and discussion

We conclude our work presented in this chapter. Additionally, we also discuss the choice of feature and it's impact on the performance.

Table 3.3: A performance comparison between PC-DARTS models and competing state-of-the-art systems reported in the literature. Results for the ASVspoof LA evaluation partition.

Systems	**Features**	**min-tDCF**	**EER**	**Params**
Res2Net [95]	CQT	0.0743	2.50	0.96M
Res2Net [95]	LFCC	0.0786	2.87	0.96M
PC-DARTS $(16, 64)$	LFCC	0.0914	4.96	7.51M
PC-DARTS $(4, 16)$	LFCC	0.0992	5.53	0.14M
LCNN [22] [96]	LFCC	0.1000	5.06	10M
LCNN [22] [96]	LPS	0.1028	4.53	10M
LFCC-GMM [15]	LFCC	0.2116	8.09	-
Res2Net [95]	LPS	0.2237	8.78	0.96M
CQCC-GMM [15]	CQCC	0.2366	9.57	-
Deep Res-Net [97]	LPS	0.2741	9.68	0.31M

3.5.1 Chapter summary

In this chapter, we present what we believe to be the first successful implementation of differentiable architecture search (DARTS) for voice spoofing detection. We demonstrate that partially connected differentiable architecture search (PC-DARTS) can effectively learn deep neural network architectures from a set of candidate operations. With PC-DARTS, architectures can be optimised using backpropagation and hardware acceleration, allowing for the automatic learning of even complex convolutional and residual networks.

The resulting models show competitive performance compared to the state-of-the-art. Our top-performing model achieves a min-tDCF of 0.09 for the ASVspoof 2019 Logical Access database, surpassed only by a Res2Net system by a minimal margin. This notable outcome is produced by a network whose architecture and parameters are learned automatically, rather than through extensive manual optimisation. Our second-best system which achieves a min-tDCF of 0.1, has 85% fewer parameters than the best-performing Res2Net system. These results encourage further exploration of PC-DARTS.

3.5.2 Discussion on the input feature

In our study, we chose Linear Frequency Cepstral Coefficients (LFCC) as our primary input feature because it outperformed other features like Linear Frequency Bank (LFB) and STFT spectrogram in our tests, even though these features were compared using the same PC-DARTS model size, with consistent settings for FFT points, window size, shift, and masking of 12 frequency channels.

Our goal with PC-DARTS was to identify optimised architectures for spoofing detection tailored to the chosen input feature, as we believe there's an optimal neural network architecture for each individual that best extracts and learns from the voice signal's cues and artefacts. However, the necessity of manually selecting input features complicated our experiment and diverged from our initial aim to streamline the development of anti-spoofing models. Moving forward, we plan to eliminate the need for choosing input features by adapting the PC-DARTS algorithm to a fully end-to-end approach, allowing the network to directly process raw waveforms.

Our initial goal in using the PC-DARTS method was to automatically discover the most efficient network architectures for spoofing detection, based on any given input feature. This approach is based on the premise that an optimal neural network architecture exists for extracting and learning from the deep representations in voice signals. However, the need to manually select features made our experiment more complex and deviated from our aim to simplify the design of anti-spoofing models. In the next chapter of this thesis, we plan to address this issue by adapting the PC-DARTS algorithm to an end-to-end (E2E) approach, which means the network will process raw audio waveforms directly, eliminating the need to choose input features manually.

Chapter 4

End-to-end neural architecture search

In this chapter, we extend our work in neural architecture search forward to a fully end-to-end (E2E) architecture. Our motivation stem from the report in the literature of successful applications of E2E models to anti-spoofing. These models operate directly upon the raw audio waveforms, and at the time that this work was conducted, were starting to outperform the more traditional deep learning approaches which operate upon hand-crafted features. This chapter presents our efforts to automatically learn the network architecture for a speech deepfake and spoofing detection system, while jointly optimising the network components and parameters, such as the initial convolutional layer that processes raw signal inputs.

Section 4.1 presents our motivation of the work. Section 4.2 describes our raw differentiable architecture search algorithm. Experiments and results are described in Section 4.3 and Section 4.4 respectively. Section 4.5 presents the summary of the work.

The work presented in this chapter was published in:

Wanying Ge, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Raw Differentiable Architecture Search for Speech Deepfake and Spoofing Detection**," in *The ASVspoof 2021 Workshop (INTERSPEECH Satellite Workshop)*.

4.1 Motivation

Our research group has investigated E2E, automatically learned network architectures in the past [83]. A crucial aspect of this research involves working directly with raw signals. By utilising a collection of sinc-shaped filters, a recent work [60] operates on the raw audio waveform through time-domain convolution, while the remaining network components are optimised in a conventional manner. The results reveal that systems with automatically learned features can perform competitively and can be complementary to those employing hand-crafted features. Although these findings are promising, performance improvements remain relatively modest. Despite the focus on E2E learning of both features and classifiers, the network *architecture* of [60] remains hand-crafted [60]. This is also true for all E2E solutions proposed to date [21, 58, 98].

In the previous chapter, we have investigated automated methods for learning network architecture. We use a particular version of differentiable architecture search [86], namely partially-connected differentiable architecture search (PC-DARTS) [30]. The architecture search is carried out using two primary network components called cells. Cells are characterised by both architectural parameters and network parameters, which are jointly optimised during the first of two phases known as the *architecture search* stage.

We demonstrated [99] that PC-DARTS can learn more compact models that remain competitive with state-of-the-art solutions. As the pioneering effort to investigate differentiable architecture search for anti-spoofing, this research was conducted using hand-crafted features. Our work presented in this chapter, therefore, aims to merge architecture search with fully E2E learning. In this work, we introduce Raw PC-DARTS, the first E2E speech deepfake and spoofing detection system which operates directly on the raw waveform while facilitating the joint optimisation of both network architecture and network parameters.

4.2 Raw differentiable architecture search

In this section, we present an overview of the proposed Raw PC-DARTS method. The model architecture is outlined in Table 4.1. We discuss the front-end sinc filter bank and the implementation of filter masking in Section 4.2.1. The back-end classifier design and base cell structure are introduced in Section 4.2.2. Finally,

Table 4.1: The proposed network structure. Each cell receives outputs of its two previous cells/layers. Conv(k, s, c) stands for a convolutional operation with kernel size k, stride s and output channel c. BN refers to batch normalisation.

Layer	Input:64000 samples	Output shape
Sinc Filters	Conv(128, 1, 64) Maxpooling(3) BN & LeakyReLU	(21290, 64)
Conv_1	Conv(3, 2, 64) BN & LeakyReLU	(10645, 64)
Normal Cells	{ BN & LeakyReLU Operations Maxpooling(2) } × 2	(2661,256)
Expand Cell	BN & LeakyReLU Operations Maxpooling(2)	(1330, 512)
Normal Cells	{ BN & LeakyReLU Operations Maxpooling(2) } × 2	(332, 512)
Expand Cell	BN & LeakyReLU Operations Maxpooling(2)	(166, 1024)
Normal Cells	{ BN & LeakyReLU Operations Maxpooling(2) } × 2	(41, 1024)
GRU	GRU(1024)	(1024)
Embedding	FC(1024)	(1024)
Output Score	P2SActivationLayer(2)	(2)

we present the extraction of embeddings and the loss function in Section 4.2.3.

4.2.1 Sinc filters and masking

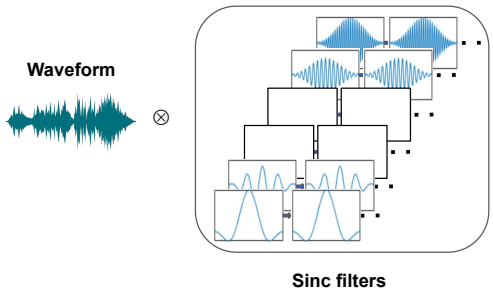


Figure 4.1: An illustration of sinc filter masking, where we use totally 6 convolutional channels as sinc filters and the 3rd and 4th filters are masked.

The input waveform of the network is fixed to a duration of 4 seconds (16000×4 samples) through either concatenation or truncation of the original audio data. Feature extraction is carried out using a collection of C sinc filters [59]. Each filter operates time-domain convolution on the input waveform. The impulse response for every filter is determined as follows:

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n) \tag{4.1}$$

where f_1 and f_2 are the cut in and cut off frequencies, and $sinc(x) = sin(x)/x$ is the sinc function. Both f_1 and f_2 can be initialised to any given frequency scale, and can set as learnable or fixed model parameters in our experiment.

During network training, we randomly mask a number of the sinc filters. This technique is similar to channel drop-out [100, 101] and frequency masking [41, 77, 102] and promote the learning of more generalised representations. In practice, sinc filters within the range $[C_1, C_2)$ are set to zero (masked), where C_1 represents the first randomly selected masked filter and $C_2 = C_1 + f$. The number of masked filters, f, is selected from a uniform distribution $[0, F)$, with F being a predefined maximum value. Once f is generated, C_1 is chosen from a uniform distribution $[0, C - f)$. And illustration of this operation is shown in Figure 4.1.

4.2.2 Search space and cell architectures

Different to the method mentioned in the previous chapter [99], where input features are treated as 2D images, Raw PC-DARTS performs operations directly on the raw time-domain waveform. As a result, the search space $\mathcal{O}$ is designed using 1D convolutional operations including: standard and dilated convolutions with kernel sizes {3, 5}; max pooling and average pooling with kernel sizes {3}; skip connections; no connections.

The original DARTS method searches for the architectures of two cell types: a normal cell and a reduction cell. The model is constructed by sequentially stacking these cells, with reduction cells positioned at $\frac{1}{3}$ and $\frac{2}{3}$ of the total network depth. Once again, normal cells maintain the feature map dimensions, while reduction cells reduce the dimensions by half and double the number of channels. A global average pooling layer is subsequently utilised after the stacked network to extract embeddings.

This stacked cell design is effective for spectro-temporal representations, as their dimensions are similar to those typically used in image classification tasks where DARTS was first applied [103, 104]. However, for speech classification tasks and raw waveform-based solutions, the feature dimension remains large at the stacked cell output, and the use of global pooling leads to significant information loss. Although a greater number of reduction cells could be manually added to reduce the feature dimension, this would undermine the goal of automated architecture search. Moreover, each additional reduction cell doubles the number of channels, resulting in increased computational complexity and GPU memory demands.

To address this issue in Raw PC-DARTS, we apply max pooling to each cell output, reducing the feature dimension by half. This simple yet effective solution helps the model learn a more compact, high-level representation without increasing the number of channels, thus reducing computational complexity and GPU memory requirements. An added advantage is that the same architecture depth and initial number of channels can be employed for both architecture search and training from scratch stages. Consequently, the so-called *depth gap* [105, 106] is avoided, where searched operations may not fit the deeper network in the second stage due to depth mismatch between architecture search and training from

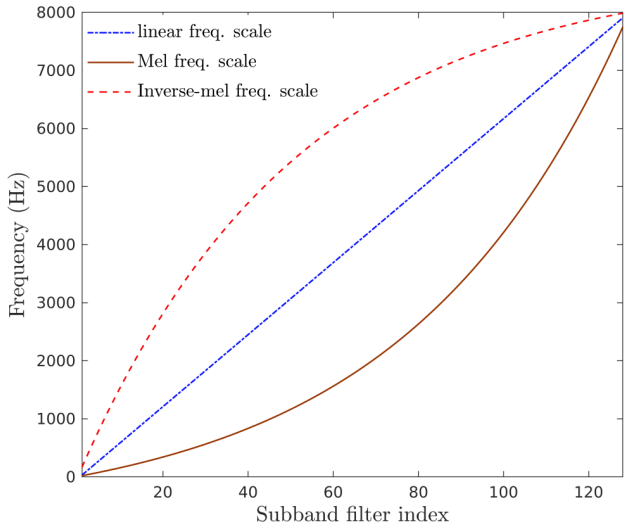


Figure 4.2: Illustration of frequency scales used to initialise sinc filters, reproduced from [2].

scratch stages. As a result, the cells used in Raw PC-DARTS are referred to as *normal* and *expand* cells. Both cells halve the input feature dimension, while only the expand cell doubles the number of channels. Expand cells are placed at the same network depth as reduction cells in the original DARTS approach.

4.2.3 Embedding extraction and loss function

The frame-level representations generated by the final cell are fed into a gated recurrent unit (GRU) layer to obtain utterance-level representations. These representations are then fed into a fully connected layer which extracts the embeddings. We use mean-square error (MSE) for P2SGrad [26] as the loss function. First, an activation layer calculates the cosine distance $\cos\theta$ between the input embedding and the class weight. As in [107], this step is hyperparameter-free, reducing the sensitivity of margin-based softmax to its scale and angular margin parameter settings and yielding relatively consistent results. The network loss is the MSE between $\cos\theta$ and the target class label. Scores used for performance evaluation correspond to $\cos\theta$ for the bona fide class.

4.3 Experimental setup

We follow the same database partition and metrics as in the previous chapter. For the sinc filters, same to [60], we explore three frequency scales as shown in Figure 4.2: Mel, inverse-Mel, and linear. For each scale, we examined two scenarios: fixed and learnable. In the fixed setting, scales remain constant throughout both the architecture search and train from scratch stages. On the other hand, learnable scales are similarly initialised, but their configuration is updated during the architecture search stage. However, they remain fixed during the train from scratch stage. Additionally, we evaluated a randomly initialised, learnable convolution block, referred to as Conv_0, as an alternative to sinc filters. The kernel size, stride and output channel number for the Conv_0 system align with those used in systems incorporating sinc filters. We set the maximum number of masked filters to $F = 16$.

In identical fashion to the approach described in Chapter 3, we maintain a fixed number of nodes $N = 7$ in each cell and limit the inputs for intermediate nodes to 2. Our models consist of 8 cells, including 6 normal cells and 2 expand cells, with an initial channel number of $C = 64$ in both stages. We conduct 30 epochs of training during the architecture search. The first 10 warm-up epochs involve updates to network parameters only, while both architecture and network parameters are updated during the following 20 epochs. We set the batch size to 14 and employ Adam optimisation [93] for learning. Architecture parameters are adjusted using a learning rate of 6e-4 and a weight decay of 0.001, while the network parameters utilise a learning rate of 5e-5. We implement partial channel selection with a value of $K_C = 2$. In the train from scratch stage, all models are trained for 100 epochs using a batch size of 32. An initial learning rate of 5e-5 is gradually reduced to 2e-5, following a cosine schedule. All models in the experiment are trained using the same random seed and executed on a single NVIDIA GeForce RTX 3090 GPU. The architecture search stage takes around 21.5 hours, while the train from scratch stage requires approximately 9.5 hours.

4.4 Results

First, we present a series of experiments evaluate Raw PC-DARTS performance with varying first layer sinc filter scales in Section 4.4.1. Subsequently, in Sec-

Table 4.2: EER results for the ASVspoof 2019 LA database, evaluation partition. Results shown for different Raw PC-DARTS setups using different first layer sinc scale initialisation.

Type	Fixed		Learnable	
	min-tDCF	**EER**	**min-tDCF**	**EER**
Mel	0.0517	1.77	0.0899	3.62
Inverse-Mel	0.0700	3.25	0.0655	2.80
Linear	0.0926	3.29	0.0583	2.10
Conv_0	×	×	0.0733	2.49

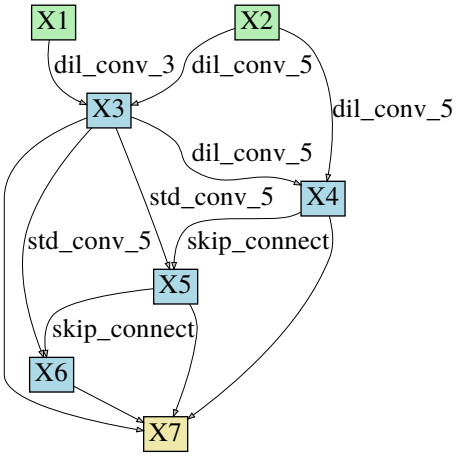
tion 4.4.2, we compare performance to existing state-of-the-art approaches. Lastly, we provide an analysis of complexity in Section 4.4.3 and an analysis of generalisability by examining performance stability across diverse spoofing attacks in Section 4.4.4.

4.4.1 Raw PC-DARTS with different sinc scales

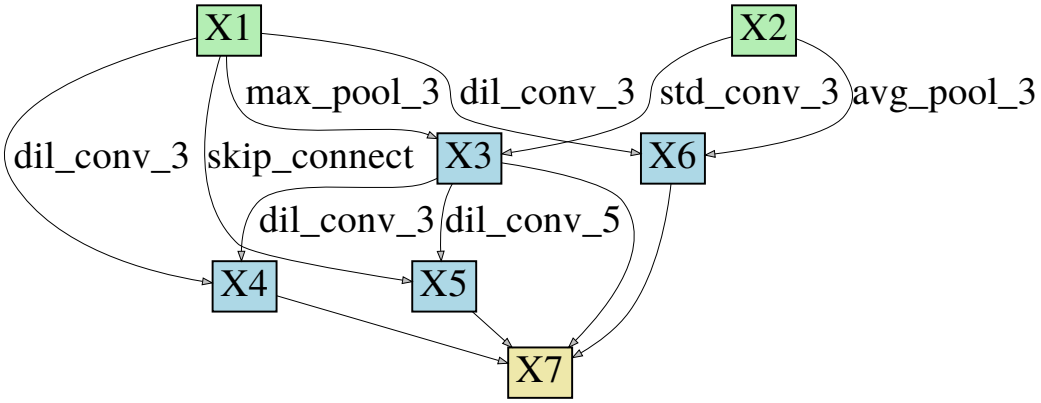
Table 4.2 shows results in terms of min t-DCF and EER for the ASVspoof 2019 LA evaluation partition. Results are presented for four distinct sinc scale configurations: Mel, inverse-Mel, linear, and randomly initialised learnable convolution blocks (Conv_0). Except for Conv_0, results for both fixed and learnable configurations are shown for each case.

The best min t-DCF (0.0517) and EER (1.77%) are achieved using fixed Mel scale sinc filters. For both inverse-Mel and linear scales, learnable configurations outperform their fixed counterparts, with the second-best result (min t-DCF of 0.0583 and 2.1% EER) obtained using a linear scale. Although the Conv_0 system delivers a commendable EER of 2.49%, its min t-DCF of 0.0733 is significantly inferior to the top-performing configurations.

The cell architectures for the optimal configuration (Mel-Fixed) are shown in Figure 4.3. We noticed that, despite the architecture parameters being randomly initialised, dilated convolution operations tend to dominate after several warm-up epochs. This observation suggests that, when applied to raw waveforms, dilated convolutions might contribute more to representation learning than other candi-



(a) Normal cell



(b) Expand cell

Figure 4.3: An illustration of the normal (a) and expand (b) cells produced by the architecture search stage for the Mel-Fixed Raw PC-DARTS configuration.

date operations within the search space. Dilated convolutions expand the receptive field [21,108,109], enabling the utilisation of more extensive contextual information to enhance performance.

4.4.2 Comparison to competing systems

Table 4.3 presents a comparison of results for the two best-performing Raw PC-DARTS systems with top-performing systems reported in the literature. The num-

Table 4.3: A performance comparison between proposed models and competing state-of-the-art systems reported in the literature. Results for the ASVspoof LA evaluation partition.

Systems	Features	min-tDCF	EER	Params	Worst attack	Worst EER
Res-TSSDNet [21]	waveform	0.0482	1.64	0.35M	A17	6.01
Raw PC-DARTS Mel-F	waveform	0.0517	1.77	24.48M	A08	4.96
ResNet18-LCML-FM [77]	LFB	0.0520	1.81	-	A17	6.19
LCNN-LSTM-sum [26]	LFCC	0.0524	1.92	0.28M	A17	9.24
Capsule Network [56]	LFCC	0.0538	1.97	0.30M	A17	3.76
Raw PC-DARTS Linear-L	waveform	0.0583	2.10	24.40M	A08	6.23
ResNet18-OC-Softmax [57]	LFCC	0.0590	2.19	-	A17	9.22
Res2Net [95]	CQT	0.0743	2.50	0.96M	-	-
ResNet18-AM-Softmax [57]	LFCC	0.0820	3.26	-	A17	13.45
ResNet18-GAT-T [110]	LFB	0.0894	4.71	-	A17	28.02
ResNet18-GAT-S [110]	LFB	0.0914	4.48	-	A17	21.74
PC-DARTS [99]	LFCC	0.0914	4.96	7.51M	A17	30.20
RawNet2 [60]	waveform	0.1294	4.66	25.43M	A18	16.30

ber of learnable parameters and the decomposed EER results for Res-TSSDNet and LCNN-LSTM-sum were obtained using open-source codes available online. The data for Capsule Network were provided by the authors of [56], and those for ResNet18-GAT and RawNet2 were supplied by the authors of [60, 110].Among the 13 systems, four process raw inputs, including the top two systems—Res-TSSDNet from [21] and the proposed Raw PC-DARTS. The fourth raw waveform-based system is RawNet2 from [60], which employs a first layer of sinc filters, a GRU and a fully connected layer for embedding extraction.

These findings demonstrate the competitiveness of raw waveform-based solutions and indicate that automatically learned cell architectures can achieve performance levels similar to or better than hand-crafted designs.

4.4.3 Complexity

Table 4.3 presents the number of network parameters for the displayed systems in column 5 (where such numbers are available). Both top-performing Raw PC-DARTS architectures possess over 24M parameters. In the Mel-Fixed configuration, 77% (18.89M) of the learnable network parameters are attributed to GRU layers, while only 18% (4.52M) are associated with stacked cells. The RawNet2

system, which also employs a GRU, has more than 25M parameters. Other systems feature significantly fewer parameters, such as the leading Res-TSSDNet system with 0.35M parameters. It utilises ResNet-style 1D convolution blocks and three FC layers, excluding GRUs. Dilated convolutions contribute to controlling network complexity while expanding the receptive field [21]. Although the LCNN-LSTM-sum system [26] incorporates two bidirectional LSTM layers, typically computationally demanding, a hidden size of 48 helps maintain the lowest complexity among all showcased systems. The increased complexity of the Raw PC-DARTS architecture is a current limitation; however, it may be a justifiable trade-off, considering the learning and optimisation process involves relatively little human effort.

4.4.4 Worst case scenario

Generalisation has been a important point in anti-spoofing research since the founding of ASVspoof in 2013. It is well-established that even top-performing systems can struggle to detect the full range of spoofing attacks [111]. Therefore, there is an interest to minimise not only pooled performance but also performance in the so-called *worst-case scenario*, which, for the ASVspoof 2019 LA database, is typically the infamous A17 attack.

Columns 6 and 7 of Table 4.3 display the worst-case attack and corresponding EER for each system. Notably, systems operating on raw waveform inputs exhibit a distinct advantage. The Res-TSSDNet [21] and both Raw PC-DARTS systems have among the lowest worst-case EERs. This finding suggests that waveform-based systems can capture discriminative artefacts which may be overlooked by systems relying on hand-crafted inputs. If an adversary were to identify and exploit only the attacks to which a system is most vulnerable, Raw PC-DARTS countermeasures would provide the second-best protection among all competing systems.

4.5 Conclusion and discussion

We first present the summary of our work on Raw PC-DARTS. We then discuss our observation from experiments reported both with PC-DARTS and Raw PC-DARTS, as well as from other literature, specially on the impact of feature

calculation towards system performance.

4.5.1 Chapter summary

Built on the PC-DARTS network presented in Chapter 3, we introduced an end-to-end differentiable architecture search approach for speech deepfake and spoofing detection, namely Raw PC-DARTS. Unlike PC-DARTS where only the network architecture is automated generated through architecture searching, we demonstrated that all components of a deep network model, including pre-processing operations, network architecture, and parameters, can be automatically learned from raw waveform inputs, resulting in a system which is competitive with the state-of-the-art.

Although the best performance is achieved using a fixed front-end rather than a learnable configuration, the latter is only slightly behind, and both systems deliver among the best performance reported to date for the ASVspoof 2019 logical access database. However, the use of gated recurrent units makes the resulting models significantly more complex than competing systems, potentially exhibiting redundancies. While reducing redundancy is possible and the results in this chapter show the potential of learned architectures, further work is necessary to address complexity, especially when computational capacity is limited and a design criterion, such as for embedded applications. One potential direction for future research is to explore replacing gated recurrent units, which have millions of parameters, with concatenated fully connected layers that possess orders of magnitude fewer parameters.

4.5.2 Discussion on system performance and behaviours

We observe that Raw PC-DARTS solutions generalise better to unseen spoofing attacks than the previous PC-DARTS solutions. Performance for the worst-case A17 attack is notably superior to that of many competing systems. This trend is not isolated; we've noticed that systems relying on spectro-temporal features generally struggle with the A17 attack, whereas certain end-to-end (E2E) models, including our Raw PC-DARTS, manage to detect it more effectively than other attacks in our test set. Changes of such behaviour appear to be directly linked to the E2E nature of these models, which operate without the need for manually

crafted features.

We plan to investigate the information or cues missed by handcrafted solutions but successfully captured by fully learned solutions in the next chapter. By understanding these aspects, we may be able to combine the benefits of both approaches, further enhancing reliability while maintaining complexity.

Chapter 5

Towards explainability in voice anti-spoofing

Unlike the previous chapters, which concentrate on system design with the primary aim of enhancing spoofing detection performance, the focus in this chapter shifts to exploring explainability. This change in research direction stems from various factors, including observed discrepancies in model behaviour and performance, as noted in previous chapters as well as by other studies [27, 60]. Additionally, the link between spoofing detection and security necessitates a broader approach, prioritising not just performance metrics like the equal error rate (EER), but also emphasising reliability and trustworthiness. Consequently, we embarked on pioneering studies into explainability, the processes and outcomes of which are detailed in this chapter.

In Section 5.1, we elucidate—and to some extent justify—our shift in focus from performance metrics to explainability with two examples. Section 5.2 outlines the methodology we employed to gain insights into explainability, specifically SHapley Additive exPlanations (SHAP). Examples of SHAP visualisation are shown in Section 5.3. The subsequent Sections 5.4 and 5.5 delve into the experimental setup and our analytical process. Finally, Section 5.6 provides a comprehensive summary.

The work presented in this chapter was included in:

Wanying Ge, Jose Patino, Massimiliano Todisco and Nicholas Evans, "**Explaining Deep Learning Models for Spoofing and Deepfake Detection with SHapley Additive exPlanations**," in *Proc. ICASSP 2022*.

Wanying Ge, Massimiliano Todisco and Nicholas Evans, "**Explainable Deepfake and Spoofing Detection: An Attack Analysis Using SHapley Additive exPlanations**," in *The Speaker and Language Recognition Workshop*.

5.1 Examples that call for explainability in anti-spoofing community

Though significant advancements have been achieved in the domain of spoofing and deepfake detection recently, the research community still faces the challenge of understanding the rationale behind a classifier's output. This section illustrates this challenge through two examples. These instances highlight what is often misconceived as a 'trick' or 'shortcut' in performance-driven system design, but in reality, they underscore our incomplete grasp of the task at hand. Furthermore, these examples underscore the need within the anti-spoofing community for tools that aid in explaining and understanding the behaviour of countermeasure systems.

5.1.1 Impact of input feature selection to system performance

The previous two chapters detailed our efforts to automate network architecture design for anti-spoofing. In Chapter 3, our initial approach yielded respectable results, albeit not surpassing state-of-the-art systems. Furthermore, when the algorithm was applied in a fully end-to-end (E2E) manner, as discussed in Chapter 4, we observed an improvement in performance. The systems in both chapters were developed using the same architecture search algorithm within similar search spaces. These spaces contained identical operations but were adapted for processing one-dimensional or two-dimensional feature maps. While changes in other network modules, such as additional pooling layers and GRU layers, may contribute to this performance enhancement, the choice of different input features could also play a significant role.

Further analysis in Section 4.4.4 and results in Table 4.3 indicate that input features significantly influence system performance, particularly in worst-case scenarios. This observation aligns with findings reported in [60], where the A17 attack in the ASVspoof 2019 LA evaluation set was more effectively detected using raw waveform inputs, as opposed to traditional, hand-crafted spectro-temporal speech

representations. This, along with our findings in Chapter 4, suggests that different forms of feature representations of the same speech signal might reveal spoofing artefacts in varied ways. This phenomenon is not isolated; numerous studies have also observed that even when the same system is applied to the same evaluation dataset, detection performance can vary significantly with different input feature representations [22, 26]. For instance, [26] reports that the EER of the same light convolution network (LCNN) on the ASVspoof 2019 LA evaluation partition varied drastically, from 14.86% with linear filter bank (LFB) inputs to 2.99% with linear frequency cepstral coefficient (LFCC) inputs.

5.1.2 Impact of non-speech intervals to system performance

The anti-spoofing task involves detecting differences between bona fide (genuine) and spoofed acoustic artefacts in speech, akin to the process employed by humans. Yet, it has been observed that some deep neural networks (DNNs) utilise variations in the non-speech intervals of voice signals to accomplish this task [27, 29]. Typically, bona fide utterances may include zero-valued silence segments, a feature often absent in spoofed utterances [29]. This disparity can inadvertently become a cue for distinguishing between bona fide and spoofed classes, as most deep-learning-based solutions, being supervised, enhance classification performance by leveraging information from the entire input utterance, without constraints on the specific intervals used.

Furthermore, some DNNs are reported to exploit semantically unrelated factors, such as the duration differences in silence intervals between bona fide and spoofed inputs, for spoofing detection [27]. While bona fide data can be naturally collected, comprising speech with intermittent silence, spoofed data, especially that generated by text-to-speech (TTS) attacks, often exhibits more varied non-speech duration. Although this difference isn't always a definitive indicator for spoofing detection, certain DNNs might partially rely on this aspect and lose robustness when non-speech intervals are eliminated during preprocessing steps like voice activity detection.

5.2 SHapley Additive exPlanations (SHAP)

In human perception, terms describing spoofed or artificial voices often revolve around phrases like 'doesn't sound like someone' or 'sounds fake,' indicating a reliance on nuances in spoken words rather than non-speech or its duration. This is natural, considering that silences—sounds with amplitudes too low to be perceived—are typically beyond human auditory perception. Initially, artificially designed spoofing countermeasures were assumed to function similarly to human perception. However, specially designed experiments have revealed discrepancies in this assumption [29,65]. Since researchers only have access to the network's output, these experiments are crafted to observe output differences when the network is trained and evaluated with varying data sets.

The calculation of output scores of deep-learning-based CMs is indirect and obscured by the depth and complexity of DNNs. Feature attribution methods like Local Interpretable Model-agnostic Explanations (LIME) [66], Gradient-weighted Class Activation Mapping (Grad-CAM) [63], and SHapley Additive exPlanations (SHAP) [68] provide insights into the connection between DNN inputs and outputs. In the remainder of this section, we introduce one of the methods, SHAP, and demonstrate its application in the context of anti-spoofing.

SHAP values, which can possess both positive and negative values, indicate the relative significance or insignificance of a specific feature in relation to a classifier output. Given a prediction function or model, $f(x)$, and a feature subset $S \subseteq F$ where F represents the complete set of features, the SHAP value ϕ_i is obtained using two models, one containing feature i and the other excluding it. For an arbitrary input x_S, predictions from both models are compared as follows:

$$\delta_i(S) = f_{S \cup i}(x_{S \cup i}) - f_S(x_S) \tag{5.1}$$

Here, $f_{S \cup i}$ denotes the model trained on the feature subset with the addition of feature i, while f_S signifies the model trained on the same subset without incorporating i. Equation (5.1) is computed for all possible subsets $S \subseteq F \setminus i$ (subsets S not containing i) to evaluate the impact of not including i in the feature pool for model training. The SHAP value is then expressed as:

$$\phi_i = \sum_{S \subseteq F \setminus i} \frac{|S|!\ (|F| - |S| - 1)!}{|F|!} \delta_i(S) \tag{5.2}$$

which is a normalized average across different permutations of $S \subseteq F \setminus i$, where $|S|$ and $|F|$ represent the number of features in S and F, respectively.

To illustrate this concept with a simple example, consider an image (or later, a speech spectrogram) containing numerous pixels (the features, or later, short-time spectro-temporal magnitude estimates). Now, focus on a single pixel i in the image. If the model predictions, when trained with and without the inclusion of i, show no difference, then pixel i has little specific relevance to the output. However, substantial differences between the two predictions imply that pixel i is relatively informative and has an influence on the model output.

For complex models like DNNs represented by $f(\mathbf{x})$, calculating ϕ_i according to Equation (5.2) is computationally demanding, as the model must be trained twice for each feature subset S. A more efficient alternative is necessary. First, the input $\mathbf{x}$ is simplified to $\mathbf{x}' = \{x'_1, ..., x'_D\}$, where $x'_i \in 0, 1$ indicates the absence (0) or presence (1) of the corresponding feature in $\mathbf{x}$, and D denotes the feature dimension. An explanation model $g(\mathbf{x}')$ is then employed to approximate $f(\mathbf{x})$:

$$f(\mathbf{x}) \approx g(\mathbf{x}') = \phi_0 + \sum_{i=1}^{D} \phi_i x'_i \tag{5.3}$$

Here, $\phi_0 = f(h_x(\mathbf{0}))$ corresponds to an all-zero input, and h_x is a mapping function that transforms $\mathbf{x}'$ into $\mathbf{x}$, i.e., $\mathbf{x} = h(\mathbf{x}')$. The model output is subsequently approximated by the sum of SHAP values associated with features for which $x'_i = 1$. $g(\mathbf{x}')$ is trained to approximate the original network output $f(\mathbf{x})$, and the coefficients ϕ_i of model $g(\mathbf{x}')$ are utilised in place of the actual SHAP values [68, 112].

Despite these approximations, computing SHAP values for DNNs remains a complex task. DeepSHAP [68] offers an efficient technique for estimating SHAP values in deep models. Assuming feature independence and model linearity, DeepSHAP approximates missing features with their expected values, thus eliminating the need for repetitive model retraining as required in Eqs.(5.1) and (5.2). SHAP values for basic network components (linear, max pooling, activation) are then estimated. The definition in Eq. (5.3) connects SHAP with DeepLIFT [113], an additive feature attribution method. DeepLIFT multipliers can be propagated

backward to estimate SHAP values at the model input level. Comprehensive details can be found in [68, 112, 114].

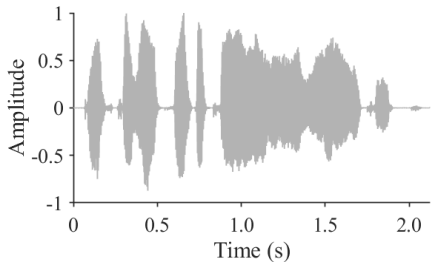
5.3 SHAP visualisation examples

In this section, we demonstrate the application of SHAP to speech data, employing two arbitrary binary classifiers for this purpose: one uses a spectro-temporal spectrogram, and the other, a raw time-domain waveform. Although other speech features, such as LFCC used in previous chapters, can also be utilised for training and often yield better detection performance, we selected these two input features due to their relatively raw representation of the voice signal, which lends itself to more intuitive visualisations.

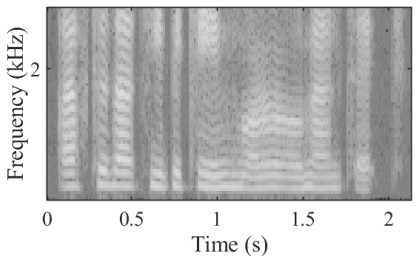
5.3.1 Spectro-temporal spectrograms

A speech waveform, $\mathbf{x}(t)$, sampled at 16 kHz is initially transformed into a short-term spectro-temporal decomposition using the short-time Fourier transform (STFT), resulting in a spectrogram, $\mathbf{X}(m, n)$, where m represents the spectral index and n is the frame index. SHAP is applied to a given (pre-trained) classifier (in this case, a spoofing detection system), treated as $f(\cdot)$ in Eq.(5.2). Each spectro-temporal point in $\mathbf{X}(m, n)$ is treated individually for every pair (m, n) as a feature, similar to i in Eq.(5.1), with the entire spectrogram considered as the full feature set F. For a given classifier, SHAP values ϕ_i in Eq. (5.2) are computed to ascertain the relative contribution of each point (m, n) in $\mathbf{X}$ to the classifier output. For a binary classifier (subsequently, a spoofing detection model), a pair of ϕ_i values is obtained, each representing support for one of the two classes (bona fide and spoofed classes). SHAP values $\phi_{(m,n)}$ can then be visualised similarly to the spectrogram $\mathbf{X}$.

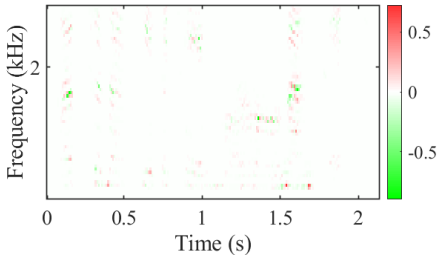
In our examples, the speech waveform $\mathbf{x}(t)$ is illustrated in Figure 5.1a, and its STFT spectrogram is shown in Figure 5.1b. We obtain SHAP values ϕ_i for both classes, with positive values indicating support for the target class and negative values for the non-target class. Focusing on *spoofing artefacts*, we always use SHAP values calculated for the spoofed class in this thesis (as shown in Figure 5.1c). Here, positive SHAP values (marked in red) indicate support for the spoofed class, while negative values (marked in green) support the bona fide class. The intensity of the



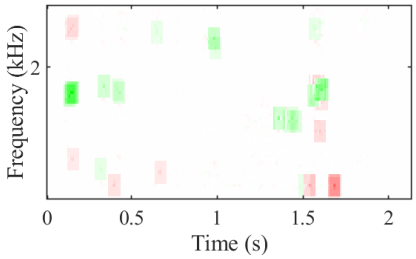
(a) Time-domain waveform



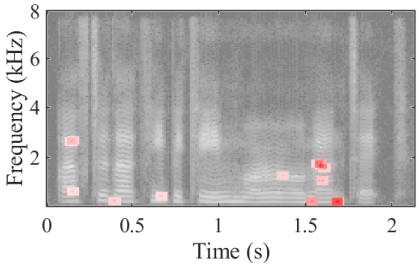
(b) Spectro-temporal STFT spectrogram



(c) Spectro-temporal SHAP values for spoofed class



(d) A dilated version of (c)



(e) Positive SHAP value for spoofed class plotted on spectrogram

(f)

Figure 5.1: Illustration of (a) original time-domain waveform, (b) STFT spectrogram, and (c-e) obtained SHAP values for utterance LA_T_3289526 '*After that he became more romantic*' and arbitrary classifiers.

colour, ranging from darker to lighter shades, corresponds to the absolute value of ϕ_i, with darker colours denoting higher values. Features contributing minimally to the classifier's output are shown in white.

Our experiments reveal that SHAP visualisations often appear sparse across the spectrogram, as evidenced in Figure 5.1c. This sparsity suggests that only a few spectro-temporal bins in $\mathbf{X}$ significantly influence the classifier's output. In the

original spectrogram, artefacts can be challenging to discern due to their small size. To enhance visualisation without compromising precision, we also present a dilated version of SHAP values in Figure 5.1d. This example highlights the differential contributions of spectro-temporal bins to classifications: lower frequency bins tend to support the spoofed class, whereas middle-region frequency bins favour the bona fide class.

However, even with the dilated version, correlating original spectro-temporal bins with their class contributions remains a challenge, as it requires referencing the original spectrogram (Figure 5.1b) with the SHAP values (Figure 5.1d). To address this, we combine images of spectrograms with corresponding SHAP values into a single figure, plotted for the full frequency scale (shown in Figure 5.1e). In further simplifying the visual representation for clarity, we eliminate the green dots that indicate support for the bona fide class, focusing exclusively on the red dots to more clearly highlight the spoofing artefacts. This approach reveals that certain lower frequency speech regions, particularly around 0.1s, 0.5s, and 1.5s in our example, are most influential for the classifier in detecting spoofed artefacts. In subsequent experiments, we employ this visualisation technique for analysing spoofing artefacts using spectro-temporal features.

5.3.2 Raw waveform

The examples in Figure 5.2 illustrate the application of SHAP to time-domain analysis, where the classifier processes the speech waveform $\mathbf{x}(t)$. The obtained SHAP values ϕ_i are of the same dimension as the original waveform. As with the previous examples, we focus exclusively on SHAP values for the spoofed class in Figure 5.2a, marking positive values (indicative of support for the spoofed class) in red and negative values (indicative of support for the bona fide class) in green. In this instance, we observe that the SHAP values for both classes are almost symmetrical, particularly around the time stamps of 0.5s, 1.0s, and 1.7s, where we find the highest amplitude in SHAP values. Similar to the spectro-temporal visualisation in Figure 5.1c, the most influential time-domain samples are distributed sparsely throughout the utterance.

While we could employ the same visualisation technique for time-domain SHAP values as used for STFT spectrogram in Figure 5.1e, we must use additional tech-

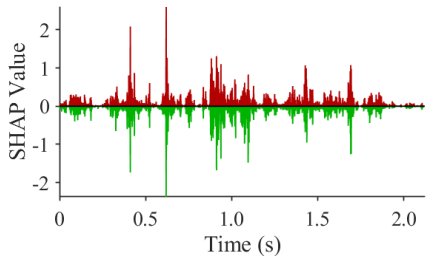
niques for further simplifying SHAP analysis in the time-domain. Because most waveform samples have lower SHAP values with low amplitude, meaning they contribute minimally to the classifier's output. These are represented by colours close to white, making the waveform difficult to discern and analyse when all values are displayed, shown in Figure 5.2b. As shown in Figure 5.2c, in a typical voice signal sampled at 16kHz, only a fraction of these samples hold high importance for the classifier's output. This fraction is approximately between 0.2% and 3% across the ASVspoof 2019 LA database.

We then prune the time-domain SHAP values to enhance clarity. Figure 5.2d illustrates the scenario where only the top 3% of SHAP values are plotted against the original waveform. While this visualisation is more discernible than Figure 5.2b, identifying specific artefacts remains challenging as they are dispersed throughout the utterance. We further refine our approach by focusing only on the top 0.2% of samples with the highest SHAP values, as shown in Figure 5.2e. This presentation is clearer, enabling easier identification of influential speech samples around 0.5s, 1s, and 1.7s which contribute most to the classifier decision. For the remainder of our experiments, we only show top 0.2% SHAP values for analysing time-domain artefacts. The same method is applied to spectro-temporal feature SHAP analysis.

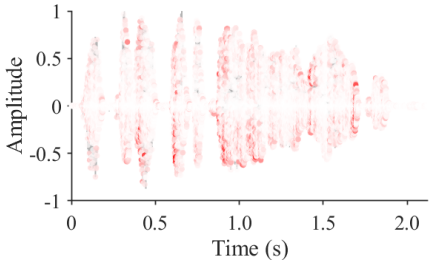
A zoomed-in version of Figure 5.2e is shown in Figure 5.2f, in which we observe a pattern similar to the findings of [115]. The data points most influential to the classifier decision are typically those with large amplitudes. This could be due to their clear difference in amplitude from nearby data points, or because high amplitude points are more unnatural in spoofed utterances compared to bona fide utterances. Regardless, this pattern suggests a common characteristic of artificial generation present in most spoofed utterances.

5.4 Classifier difference

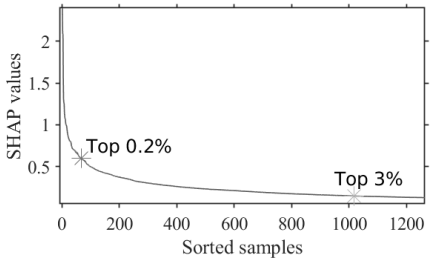
Results reported in Chapter 4, Table 4.3, indicate that the A17 attack in the ASVspoof 2019 LA evaluation set is particularly challenging to detect for countermeasures that process spectro-temporal speech representations. Interestingly, CMs that process raw waveform data do not necessarily find the A17 attack as challenging. This pattern holds true for our models as well, namely PC-DARTS introduced in Chapter 3 and Raw PC-DARTS detailed in Chapter 4. Although



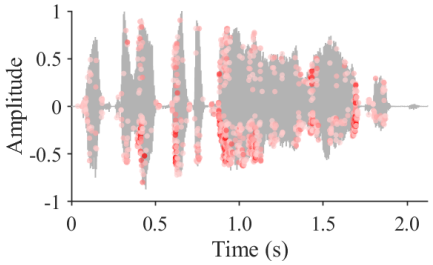
(a) Time-domain SHAP values for spoofed class



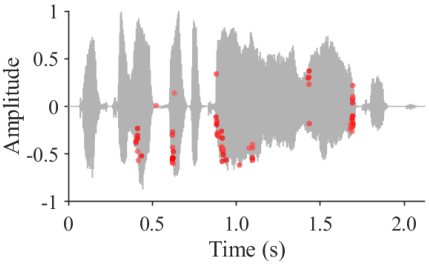
(b) Positive SHAP values for spoofed class plotted on waveform



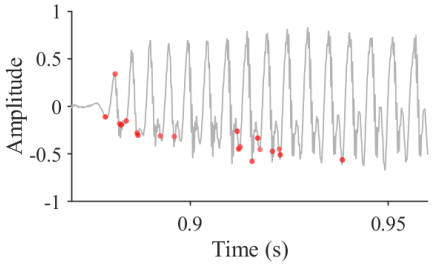
(c) Sorted positive SHAP values for spoofed class



(d) Top 3% highest SHAP values of (b)



(e) Top 0.2% highest SHAP values of (b)



(f) A zoomed in version of (e) around 0.9s.

Figure 5.2: Illustration of (a) original time-domain waveform, (b) STFT spectrogram, and (c-h) obtained SHAP values for utterance LA_T_3289526 '*After that he became more romantic*' and arbitrary classifiers.

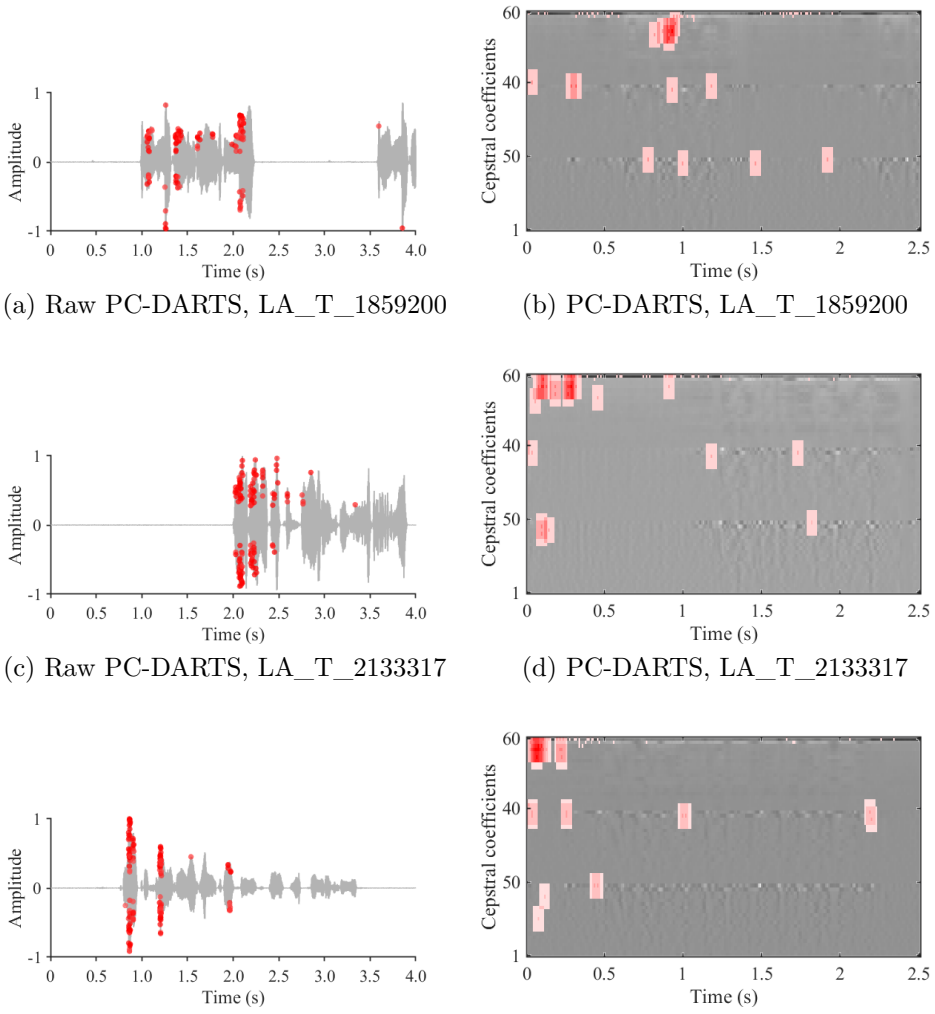
several factors contribute to the differences in system performance, results in Table 4.3 suggest that the choice of input feature significantly influences the outcome. Consequently, we set out to investigate the differences in features deemed important by the classifier.

5.4.1 Analysis with PC-DARTS and Raw PC-DARTS

Both models described in Chapters 3 and 4 were used directly, without additional training. Our PC-DARTS model, which utilises spectro-temporal Linear Frequency Cepstral Coefficients (LFCC) features, comprises 4 layers and 16 initial channels. Conversely, the Raw PC-DARTS model, which processes raw waveform inputs, is configured with 8 layers and 64 initial channels. Both models are designed to handle a fixed 4-second input, with shorter files being concatenated and longer ones being truncated. Figure 5.3 showcases SHAP visualisations for an arbitrary utterance, highlighting the time waveform and spoofed artefacts recognised by the Raw PC-DARTS model (left of Figure 5.3), and the spectro-temporal LFCC features and spoofed artefacts identified by the PC-DARTS model (right of Figure 5.3).

Observations from Figures 5.3a and 5.3c and 5.3e reveal that Raw PC-DARTS tends to focus on speech intervals with high amplitudes, typically around 1.0s, 1.5s and 2.0s in Figure 5.3a, around 2.0s to 2.5s in Figure 5.3c and around 0.9s, 1.3s and 2.0s in Figure 5.3e. Despite different example utterances and classifiers, both Figure 5.2 and Figure 5.3 corroborate the significance of vowels in waveform-based spoofing CMs. This is not unexpected, as these regions often exhibit greater energy and longer duration compared to other parts of speech.

In contrast, the PC-DARTS model, which processes LFCC features, exhibits different behaviour, as shown in Figures 5.3b, 5.3d and 5.3f. The y-axis for plots depicting cepstral representations like LFCCs and MFCCs does not represent frequency, and thus the LFCCs feature lacks the clear structure of the original spectrogram. However, by examining the x-axis, which corresponds to the time domain and aligns precisely with the waveform, we notice that PC-DARTS focuses more on the non-speech intervals. The highest concentration of red dots, which indicate the most informative intervals, are located within the first 0.5s, segments corresponding to non-speech in the original waveform. Additional experiments with other utterances consistently demonstrate that the PC-DARTS model leverages artefacts present in the non-speech intervals for spoofing detection.



(a) Raw PC-DARTS, LA_T_1859200 (b) PC-DARTS, LA_T_1859200

(c) Raw PC-DARTS, LA_T_2133317 (d) PC-DARTS, LA_T_2133317

(e) Raw PC-DARTS, LA_T_2724328 (f) PC-DARTS, LA_T_2724328

Figure 5.3: SHAP values for Raw PC-DARTS (time-domain waveform) and PC-DARTS (Linear-frequency cepstrum coefficient feature) for three utterances selected from the ASVspoof 2019 LA training partition: LA_T_1859200 '*We will do so again*', LA_T_2133317 '*I don't think the Saudis will lay down*', LA_T_2724328 '*The orchestra was already increasing the scope of its ambitions*'. The y-axis in cepstral representations does not represent frequency, and thus the LFCC feature lacks the clear structure compared to the STFT spectrogram.

5.4.2 Further analysis and discussion

We demonstrate the application of SHAP in revealing features that influence the outputs of spoofing detection models, highlighting the classifier focus across

spectro-temporal and time-domain intervals. The classifiers examined are those described in previous Chapters 3 and 4. The transition from spectro-temporal representations in Chapter 3 to raw waveform in Chapter 4 was driven by the hypothesis that direct learning from speech waveforms would more effectively capture or preserve artefacts. Our findings confirm that these classifiers address the detection task differently, indicating their reliance on various parts of the input for spoofing cues.

Particularly intriguing is that the PC-DARTS classifier uses non-speech intervals for detection, challenging the common assumption that spoofing primarily involves learning artefacts in *speech*. This raises concerns about robustness in real-world applications, especially when used alongside voice activity detection (VAD) systems, which trim non-speeches. If non-speeches are removed, PC-DARTS might lose critical detection cues, increasing the risk of false acceptances. One solution could be to replace PC-DARTS-like models with those similar to Raw PC-DARTS, which show better performance. However, as evidenced in Table 4.3, even the best-performing system, Res-TSSDNet, struggles with the A17 attack. Our hypothesis is that the artefacts in the A17 attack reside within speech intervals, which models like PC-DARTS tend to overlook. Therefore, even the top-performing systems may exhibit reduced robustness when paired with VAD.

To gain a deeper understanding, we focus on the fact that both classifiers were trained on the same ASVspoof 2019 database. Hence, different artefacts revealed in the training data might be causing the observed behaviour variations. Our next step is to determine if we can pinpoint these differences. The ASVspoof 2019 LA database comprises over 25,000 utterances. Although individual analysis of each utterance is impractical, we can still investigate commonalities. Apart from bona fide utterances, spoofed utterances are generated by six different spoofing attacks. Previous studies suggest that artificially generated utterances exhibit similar artefacts within each spoofing algorithm category [116–118]. In the following section, we report a SHAP analysis focusing on the distinct attack artefacts present in the ASVspoof 2019 LA training data.

5.5 Attack difference

In the previous experiment reported in Section 5.4, we used existing classifiers without training new ones. However, as our focus shifts to discerning differences among various spoofing attacks, it becomes necessary to adjust our choice of classifiers. This is to ensure that the differences we observe are attributable to the attacks themselves, rather than the classifiers.

Choice of classifiers – we use two Res-TSSDNets [21] for our experiments reported in this section, this is because:

1. The PC-DARTS and Raw PC-DARTS models used previously, though developed using the same algorithm, still differ in other aspects. For instance, they vary in the number of layers; the Raw PC-DARTS includes a GRU layer whereas PC-DARTS employs a pooling layer. Additionally, they utilise different loss functions during training.

2. In contrast, the Res-TSSDNet pair (1D- and 2D-Res-TSSDNet systems as proposed in [21]) feature a more uniform network architecture. Each comprises several convolutional blocks with residual connections, a global max pooling layer, and three fully-connected layers. They are designed to be efficient and trainable rapidly, producing scores for both bona fide and spoofed classes. The primary distinction between these two models lies in their treatment of input features. Specifically, the 1D-Res-TSSDNet is based on 1D convolutional operations initially applied to raw audio waveforms, whereas the 2D-Res-TSSDNet is based on 2D convolutional operations applied to magnitude spectrograms, extracted using 20ms Hamming windows with a 10ms shift and a 320-point FFT.

3. The relatively simple structure of Res-TSSDNet offers better potential for explainability. The 1D version has only 0.35M trainable parameters, and the 2D version has just 0.97M, compared to our own solutions which have significantly more, as detailed in Table 4.3.

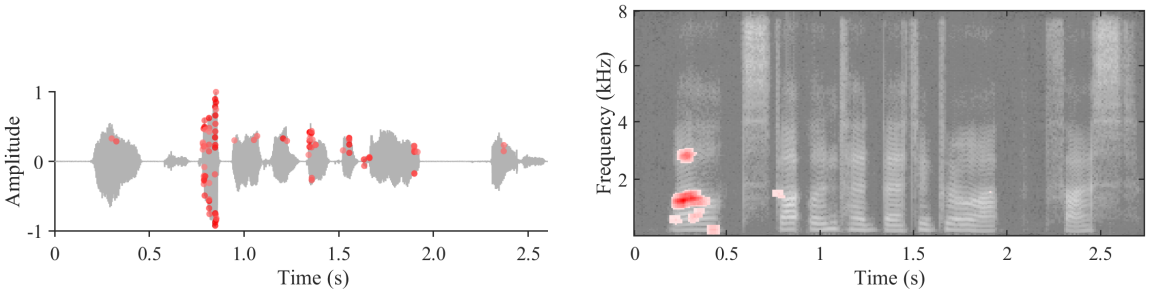
Classifier retraining – we retrain the classifiers rather than using pre-trained ones because:

1. We train the classifiers with variable, full-length utterances, rather than fixed-length inputs created through concatenation or truncation. This approach is more conducive to explainability, as it allows for a more straightforward and consistent examination of specific temporal or spectral interval influence on classifier output. Our experience suggests that concatenation can lead classifiers to interpret the same information differently in repeated short intervals, as observed in Figure 5.3a, which complicates explainability analysis.

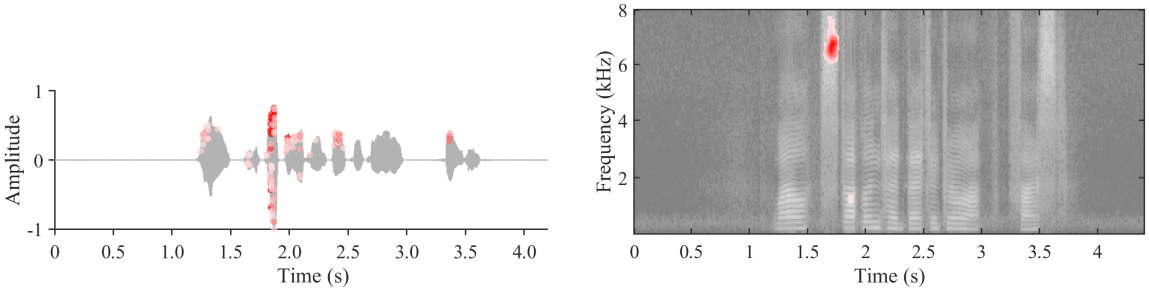
2. We retrain the classifiers using bona fide data and selected spoofed data from each specific attack, instead of the entire ASVspoof 2019 LA training set. For example, SHAP analysis for an A01 attack utterance is conducted using classifiers trained solely with A01 attack utterances. This targeted approach aids in identifying consistent artefacts associated with each attack. We acknowledge that it is impossible to fully isolate classifier behaviour from attack artefacts, as the latter are always observed using a specific classifier. By training the classifiers with matched attack utterances, we hope to observe the artefacts which are unique to each attack more clearly and consistently.

5.5.1 Experimental setup

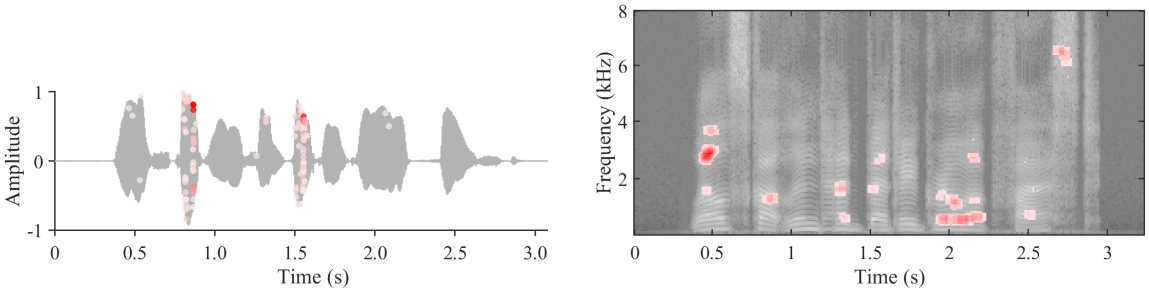
Both Res-TSSDNets are optimised by minimising the weighted cross-entropy loss between spoofed and bona fide classes. Models are trained with the Adam optimiser [93], using a learning rate of 0.001 and an exponential learning rate decay of 0.95. Variable-length input data is handled as described in [26], where training utterances of similar length are zero-padded to create uniformly-sized mini-batches. However, for experiments involving SHAP analysis rather than model updating, utterances are fed to classifiers without padding. Although using variable-length inputs does impact performance, the focus of this work is on explainability, not performance. For reference, the EERs of the 1D-Res-TSSDNet and 2D-Res-TSSDNet are 6.90% and 4.28% respectively.



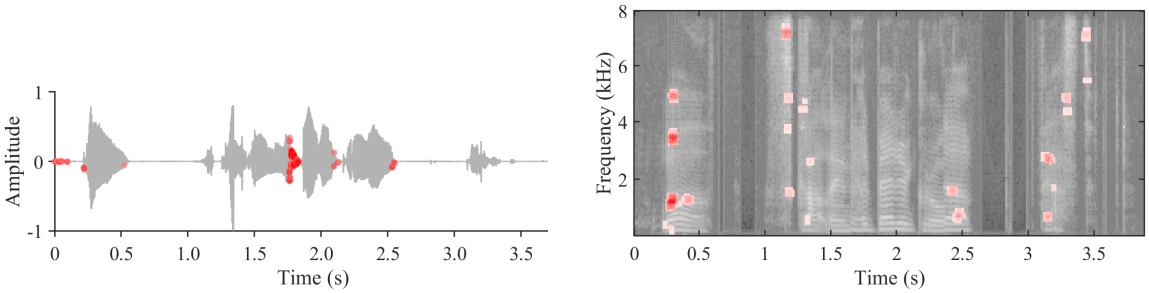
(a) A01 attack utterance LA_T_3566209



(b) A02 attack utterance LA_T_1590397



(c) A03 attack utterance LA_T_2909480



(d) A04 attack utterance LA_T_5116902

Figure 5.4: SHAP values for the A01-A04 utterance '*Well, Scotland had better grow up, fast*'.

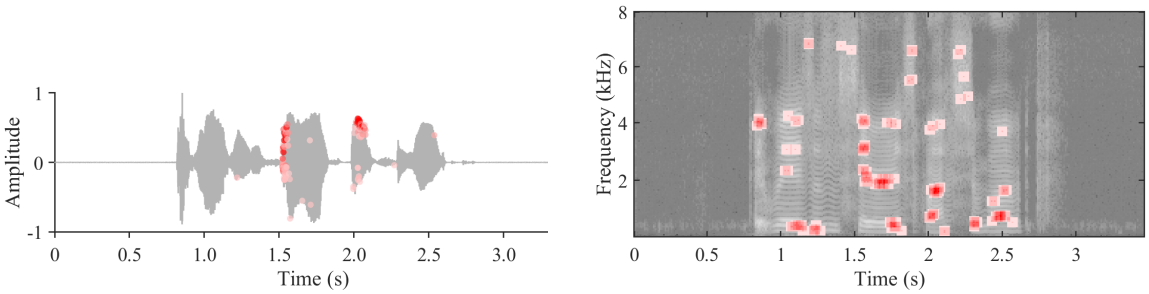
5.5.2 Results and analysis

We present an attack analysis focused on spoofing attacks contained in the ASVspoof 2019 LA training partition. This analysis includes spoofed utterances generated with 6 different algorithms, consisting of 4 TTS attacks (A01-A04) and 2 VC attacks (A05-A06). The following analyses were performed on a random selection of 100 utterances for each training attack. Example results are illustrated in Figure 5.4 and Figure 5.5, which show both waveforms and spectrograms with the corresponding superimposed 0.2% highest intensity-encoded SHAP values. The illustrated examples are specifically chosen to use the same utterance for TTS attacks and the same utterance for VC attacks. A discussion of results for each attack is presented in the following, while a summary of the principal, consistent artefacts observed in each case is presented in Table 5.1.

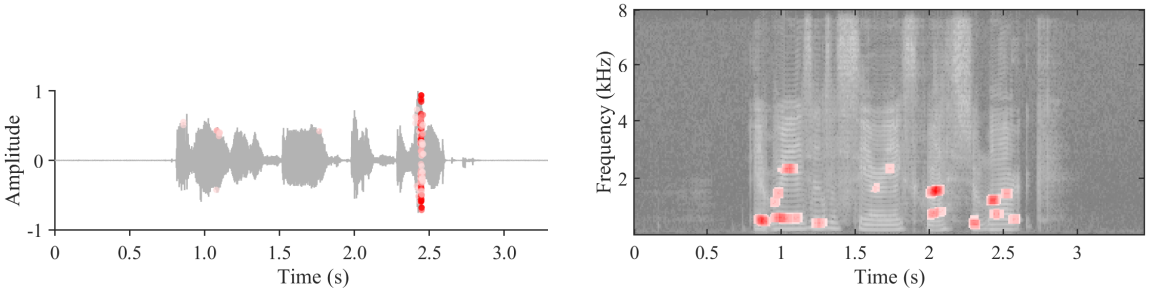
A01 is a neural network (NN) based TTS attack with a WaveNet [119] vocoder. A waveform and spectrogram with superimposed SHAP values are illustrated in Figure 5.4a. We observe differences for 1D and 2D classifiers. For the 1D classifier, we found most artefacts in vowel segments, though we could not identify a particular vowel for which SHAP values are consistently the highest. Most artefacts are found within low-frequency bands. For a substantial number of utterances, the 2D classifier identifies artefacts in the leading 0.5 seconds of speech. This might be the result of A01 attacks having a consistently shorter leading non-speech interval compared to bona fide utterances [27].

A02 is also an NN-based TTS attack, but with a WORLD [120] vocoder. Results are illustrated in Figure 5.4b. Like the A01 attack, the 1D classifier finds artefacts in vowel segments, such as the \o\ vowel in the given example. The 2D classifier finds artefacts mostly at lower frequencies and also at higher frequencies above 6 kHz. For A02 attacks and the 2D classifier, consistent artefacts are identified for the unvoiced sound \s\ segments.

A03 is a different NN-based TTS attack also with a WORLD vocoder. Results are shown in Figure 5.4c. For the 1D classifier, artefacts are found mostly in vowel segments, but are less densely distributed compared to A01 and A02 (relatively fewer dark-red points). The reason might be that artefacts in A03 attacks are located in particular samples which are different from the neighbouring ones. While



(a) A05 attack utterance LA_T_3134909



(b) A06 attack utterance LA_T_5300749

Figure 5.5: SHAP values for the A05 and A06 utterance '*It raises a serious question mark*'.

we find that artefacts lie mostly at lower frequencies in the case of the 2D classifier, we did not succeed in identifying artefacts within consistent speech segments.

A04 is a waveform concatenation TTS attack. Results are shown in Figure 5.4d. For some A04 attacks, we find artefacts to lie within leading non-speech and low energy speech segments (onsets and offsets of speech), a characteristic that differentiates A04 from the other TTS attacks. These observations may correspond to the use of waveform concatenation and may explain why such attacks generated with the MaryTTS platform [121] were initially challenging to detect [122]. The 2D classifier uses cues throughout the full spectrogram. Other consistent artefacts were found in unvoiced segments and click sounds (around 1.2 and 3.5 seconds for the given example).

A05 is an NN-based VC attack. Results are shown in Figure 5.5a. In addition to more dominant artefacts in vowel segments (around 2.0 seconds in the given example), the 1D classifier also finds consistent artefacts in lower energy voice

Table 5.1: Artefact description of attacks in ASVspoof 2019 LA train partition.

		Found artefacts	
Attack	Algorithm	Waveform	Spectrogram
A01	TTS	Vowels	Lower frequency bands, leading 0.5s
A02	TTS	Single dominant vowel	Lower & higher frequency bands, unvoiced \s\
A03	TTS	Less densely distributed in vowels	Lower frequency bands
A04	TTS	Non-speech, low energy speech segments (voice onsets and offsets)	Full spectrum, unvoiced speech, clicks
A05	VC	Voice onset, vowels	Full spectrum, higher energy formant frequencies
A06	VC	Speech distortion	Lower frequency bands

onset segments (around 1.6 seconds). Similar to the TTS A04 attack, the 2D classifier finds artefacts across the full spectrum, rather than specific sub-bands. Nonetheless, higher SHAP values correspond generally to lower frequencies and higher energy formant frequencies (around 2 and 4 kHz).

A06 is a transfer-function-based VC attack [123]. Results are shown in Figure 5.5b. For the 1D classifier, we observed temporal intervals with high SHAP values to correspond to noticeably distorted speech sounds. These seem to correspond to variations in unnaturally high velocity. Using the 2D classifier, artefacts are found mostly at lower frequencies below 3 kHz. They were not found to correspond consistently to any particular speech sounds other than high-energy segments, and neither do they correspond consistently to the distortions identified using the 1D classifier.

5.5.3 Further analysis

Among the four synthetic speech attacks (A01-A04), three are neural network-based text-to-speech algorithms (A01-A03) that exhibit artefacts primarily within vowel segments. In contrast, the fourth attack (A04), which uses a waveform concatenation approach, displays different artefacts found in lower energy segments.

These variations in artefacts may indicate the underlying spoofing attack method. We also observed distinctions between the two voice conversion algorithms. The A05 attack shows artefacts during voice onset segments and throughout the full spectrum, while the other A06 attack is associated with noticeable distortions at lower frequencies. Certain consistencies exist between artefacts produced by both voice conversion and synthetic speech attacks, particularly in low-frequency bands.

Our initiative to analyse the variation in spoofing attacks is grounded in the understanding that the spoofing detection capabilities of CMs are derived from learning patterns within the training database. As highlighted in the previous Section 5.4, there must be certain spoofed training data containing artefacts either in speech intervals or in non-speech intervals. Consequently, the CMs trained on this data learn to utilise these distinct cues, either in speech or non-speech, for spoofing detection. Our analysis in this section reveals that nearly all spoofed training data (from five out of the six attacks) predominantly exhibit artefacts within speech intervals, especially in vowels. Interestingly, the A04 attack is unique in presenting artefacts in non-speech intervals. This does not necessarily imply that the A04 attack algorithm leaves no artefacts in speech; it might be that the non-speech intervals in utterances generated by A04 are more easily detectable compared to those in its speech intervals. Nevertheless, it is evident that the presence of A04 in the training data influences some classifiers to leverage non-speech intervals as a key indicator for spoofing detection.

5.6 Conclusion, limitations and discussion

This chapter showcases the application of SHAP analysis to explain both the differing behaviour of our proposed models and the distinctive artefacts of various spoofing attacks. We first demonstrate how our models, despite being developed using the same algorithm and trained with identical data, respond differently to the same input. We then illustrate how even a single classifier can exhibit varied behaviours when trained with different subsets of the training data. This indicates that different spoofing algorithms tend to imprint distinct types of artefacts. Such variations in artefacts explain why models trained with the ASVspoof 2019 LA dataset sometimes focus on non-speech intervals, contrary to the common assumption that emphasis should be on speech intervals.

Specifically, we observed that the PC-DARTS model, as outlined in Chapter 3, tends to rely on information in non-speech intervals for spoofing detection. In contrast, the Raw PC-DARTS model from Chapter 4 primarily uses information in speech intervals for the same purpose. This distinction, which was not anticipated during the design phase of these systems, may explain the performance differences between the two models, despite their shared attributes in the architecture design. A more detailed examination of attack-specific artefacts revealed that, while most spoofed training data contain artefacts within speech intervals, it is particularly the A04 attack that leaves noticeable artefacts in non-speech intervals, leading some CMs, including PC-DARTS, to focus on these non-speeches instead of speech.

Spoofing and deepfake detection system will effectively benefit from accurately identifying and analysing artefacts, as demonstrated through our experiments. This approach aligns with the General Data Protection Regulation (GDPR), which requires that data subject have the right to obtain meaningful information about the logic involved in automated decision-making. Our use of visual analyses, including spectro-temporal and time-domain analysis, offers clearer explanations than just output scores, moving towards providing 'meaningful information'.

However, these explanations don't fully meet the criteria for being 'meaningful.' For example, a meaningful explanation in a bank loan denial would tell the applicant how to improve their application for future acceptance, such as suggesting an increase in the number of working years. In contrast, for spoofing countermeasures or speaker verification systems, showing a heat-map of feature importance is less helpful. Interpreting these maps requires audio signal processing knowledge, and even with this knowledge, users cannot modify their speech in such a detailed manner to affect future outcomes.

Another aspect of our study involves using the artefact analysis as a feature to enhance classification performance. However, our findings are based on 100 spoofed examples from each type of attack, and not all examples showed consistent artefact patterns. This inconsistency could potentially confuse the classification system or complicate the training process. Additionally, using these explanations as input for the network or a new classifier might not add new information but rather reinforce existing decisions, similar to knowledge distillation.

Addressing inconsistency was a significant hurdle in our experiments. To com-

bat this, we tailored our experimental setup, for instance, by utilising identical network structures for analysing time-domain waveforms (1D-Res-TSSDNet) and spectro-temporal spectrograms (2D-Res-TSSDNet), and training classifiers with only bona fide and spoofed utterances from specific attacks instead of the entire ASVspoof 2019 database. This approach was chosen to detect consistent artefact patterns. Despite these efforts, we recognise that factors beyond the classifier architecture and training data influence our observations. Variables such as the size or shift in feature extraction windows, network initialisation (e.g., random seed), training hyper-parameters, and even the feature attribution methods impact our observations. Our preliminary experiments confirmed that changes in these variables can complicate the observation process, affecting the classifier parameters and its ability to consistently identify spoofing artefacts.

This issue is not unique to anti-spoofing measures but is a general challenge in deep learning. Just as voice spoofing detection models exhibit varied performance based on different initialisation, hyper-parameters, or data partitions, spoofing generation models may show similar variability. This suggests that artefacts we identify as attack-specific could vary with the training conditions of the attack algorithm, potentially eluding detection by countermeasures designed to recognise them. Our attempt to verify such variability in the spoofing generation process will be detailed in the next chapter.

Chapter 6

Exploring variability in spoofing artefacts

The observation from the previous chapter – that different spoofing attack algorithms, specifically text-to-speech (TTS) and voice conversion (VC) based generation algorithms, leave distinct artefacts – is intuitive and supported by not just our findings but also those of other independent studies [22, 26, 45]. It has been noted that deep neural network (DNN) based approaches to spoofing and deepfake detection vary as much in terms of modules and architectures as they do in performance. Similar to spoofing detection methods, which are predominantly deep-learning-based today, voice generation methods also rely primarily on deep learning technology.

The work presented in this chapter explores whether just as deep-learning-based voice spoofing detection models show varied performances and behaviours under different initialisation, hyper-parameters, or training data partitions, for deep-learning-based spoofing generation models might also generate different artefacts. This variability could be significant enough that countermeasures (CMs) might fail to detect them. If this hypothesis holds true, it suggests that what we identify as attack-specific artefacts may merely be the behaviour of an attack algorithm under specific training conditions, subject to change if the model is trained differently. It also implies that an adversary could potentially circumvent detection solutions simply by altering the training parameters of a TTS or VC algorithm, even one previously used in generating spoof detection training data.

In this chapter, we aim to verify this hypothesis. The content herein also

complements the analysis presented in Chapter 5. While both chapters concentrate on differences in artefacts, Chapter 5 utilised feature attribution methods for a visualisation-based analysis, whereas in this chapter, the focus is upon analysis in terms of detection performance. Furthermore, in this chapter we set the groundwork for what follows in Chapter 7. To enhance our approach, we propose the integration of speaker verification systems with spoofing countermeasures for more robust and generalised spoofing detection.

This chapter is organised as follows: In Section 6.1 we describe spoofing attack algorithm and specific countermeasures. Section 6.2 presents the spoofing generation settings and experimental setup. Our results are presented in Section 6.3. Finally, Section 6.4 presents the summary of the work.

The work presented in this chapter was published in:

Wanying Ge, Xin Wang, Junichi Yamagishi, Massimiliano Todisco and Nicholas Evans, "**Spoofing Attack Augmentation: Can Differently-trained Attack Models Improve Generalisation?**," in *Proc. ICASSP 2024*.

6.1 Spoofing attack and countermeasures

While numerous TTS and VC algorithms have been proposed, it is impractical to analyse them all. Therefore, for the experimental work presented here, we have selected one of the most popular TTS algorithms, known as Variational Inference with adversarial learning for end-to-end Text-to-Speech (VITS) [124]. This algorithm serves as the spoofing attack in our study. Additionally, we examine three state-of-the-art spoofing countermeasures. ALL are described in the following sections.

6.1.1 VITS

VITS [124], a variational auto-encoder (VAE) [125] based TTS model, converts a phoneme sequence into a speech waveform. We chose VITS for our analysis primarily because of its efficient training procedure, which is performed end-to-end without the need for separate training of duration and acoustic models, or a neural vocoder. The high quality of synthesised speech produced using VITS can be attributed to the integration of adversarial training, normalising flow [126], and stochastic duration modelling [124].

One notable feature of VITS is its ability to produce speech data with varied tempo, intonation, and other suprasegmental attitude, even when using the same model and phoneme input. This versatility is achieved by adjusting the power of two types of random noise. The first random noise, transformed by a flow-based model [127], is used to generate discrete numbers which represent the duration of input phonemes. The second random noise comes into play when latent acoustic features are sampled through the reparameterization trick from the VAE posterior distributions [125], conditioned on the input phonemes and the generated duration. Subsequently, waveforms are generated from these latent acoustic features. By varying the power of the random noises, we can employ the same VITS model to produce speech data with diverse characteristics.

6.1.2 Countermeasures

AASIST [78] is a state-of-the-art end-to-end (E2E) spoofing countermeasure solution, utilising graph attention networks [128]. It employs a sinc-layer front-end [59], to extract feature representations from raw waveform inputs. The backend integrates both temporal and spectral representations through graph attention layers. The process is finalised with a readout operation and a fully connected output layer, which together produce detection scores.

RawNet2 [60], also an E2E model, is composed of a sinc-layer, six residual blocks, a recurrent layer with a Gated Recurrent Unit (GRU), and a fully connected output layer. The output from the sinc-layer is processed through residual blocks to extract frame-level deep feature representations. These representations are then aggregated at the utterance level by the GRU layer, culminating in the generation of detection scores by the output layer.

Self-supervised leaning with AASIST (SSL-AASIST) [79] merges the AASIST backend with front-end feature extraction using a pre-trained wav2vec2.0 model [129]. Pre-training is performed using 437k hours of bona fide speech utterances, sourced from five distinct speech databases. Together, the dataset covers 128 different languages and includes voices from over 60k speakers. SSL-based front-ends are known for their robustness to additive noise, reverberation, and other external factors [130].

The three CMs selected for our study represent the state-of-the-art in terms

of their ability to generalise across both known and unknown spoofing attacks. In our experiment, these CMs are trained to detect artefact patterns in spoofed utterances produced by the VITS model under one specific training condition. Subsequently, we assess the effectiveness of the learned CMs in detecting spoofed utterances generated by differently trained VITS models. Our objective is to determine whether variations in the training of VITS models result in distinct artefact patterns and to explore the possibility that an adversary could overcome detection solutions by merely altering the training conditions of the attack algorithm.

6.2 Experimental setup

In this section, we delineate the databases and protocols employed, alongside the specifics of the VITS training conditions, data generation procedures, and details of implementation and metrics used.

6.2.1 Databases

We utilised the VCTK database for training VITS models [131]. Given that CM architectures and hyper-parameters were originally designed and optimised using the ASVspoof 2019 database, we adhered to the same data preprocessing pipeline. The VCTK data was downsampled to 16 kHz and subjected to high-pass filtering with a cut-in frequency of 80 Hz prior to VITS model training. Speech data synthesised using VITS also maintained a sampling rate of 16 kHz. For CM training, we used a set of bona fide VCTK data and additional synthesised data generated using different VITS model configurations.

6.2.2 VITS conditions

As outlined in Table 6.1, we prepared four distinct datasets (V1, V2, V3, and V4) by modifying three VITS model configuration parameters. Additionally, for the V1 VITS model, four supplementary sets (V1.2, V1.3, V1.4, and V1.5) were generated by altering two extra hyper-parameters. These variations included adjustments in ① the training data, ② the number of Mel channels, and ③ the random seeds for initial model parameterization. Generation conditions were varied based on the noise standard deviation for ④ acoustic feature generation and ⑤ duration, as detailed in Sec. 6.1.1.

Table 6.1: VITS training and generation settings across different sets ('-' indicates identical settings to V1). Table reproduced from [3]

	Training			**Noise std. in generation**	
Set ID	Train set	#. Mel chan.	Seed	For acoustic feat.	For duration
V1	set-1	80	seed-1	0.667	0.8
V2	-	40	-	-	-
V3	set-2	-	-	-	-
V4	-	-	seed-2	-	-
V1.2	same VITS model as V1			-	-
V1.3	same VITS model as V1			0.1	-
V1.4	same VITS model as V1			-	0.1
V1.5	same VITS model as V1			0.1	0.1

For ①, we generated data using both 80-band and 40-band Mel-scaled spectrograms. For ②, a random selection of 3,000 utterances from the VCTK dataset formed the bona fide partition of the CM training set, with the remainder split into two subsets for VITS training. For ③, two distinct random seeds were employed. The sets V1.2 to V1.5 were generated using the V1 VITS model with various noise standard deviation values, as listed in Table 6.1. Notably, V1.2, despite sharing the same model and noise standard deviation as V1, consists of different data due to distinct noise values. We used datasets V1 to V4 for CM training and testing, and sets V1.2 to V1.5 solely for testing purposes.

6.2.3 Implementation and metrics

All VITS networks were trained with an initial learning rate of 2×10^{-4} and a scheduling factor of $0.999^{\frac{1}{8}}$ per epoch. The batch size was set at 50. Training was performed using two NVIDIA GeForce RTX 3090 GPUs and continued for 300k steps. CMs were trained using publicly available codes and their default settings. To evaluate spoofing detection performance, we employed the equal error rate (EER) as the metric.

Table 6.2: CM performance in terms of the EER (%) in different training and testing conditions.

	Trained on V1			**Trained on V2**			**Trained on V3**			**Trained on V4**		
Tested on	AASIST	RawNet2	SSL-AASIST	AASIST	RawNet2	SSL-AASIST	AASIST	RawNet2	SSL-AASIST	AASIST	RawNet2	SSL-AASIST
V1	0	0	0	0	13.27	0.04	0.03	6.17	1.37	0.27	12.60	0.57
V2	0.50	6.27	0.07	0	0.03	0	0.67	8.70	0.47	0.67	11.23	0.13
V3	2.43	8.50	0.03	2.20	18.00	0.10	0	0	0	1.73	10.60	0.07
V4	1.20	7.93	0	0.57	15.93	0.07	0.13	5.87	0.13	0	0.13	0
V1.2	0	0.67	0	0	13.03	0.30	0	5.47	1.40	0.23	12.47	0.60
V1.3	0	0.03	0	0	7.20	0.57	0	2.00	2.03	0.07	6.2	1.03
V1.4	0	0.93	0	0.03	12.63	0.33	0.03	7.27	1.80	0.33	15.03	1.07
V1.5	0	0.10	0	0	6.63	0.83	0.03	2.10	2.80	0.10	7.80	1.33
Pooled	0.77	3.73	0.01	0.50	11.49	0.37	0.16	5.03	1.50	0.57	10.11	0.63

6.3 Results and discussion

Results in Table 6.2 present EER estimates for each CM under various matched and mismatched training (row 1) and testing (column 1) conditions. When training and testing data are matched (as seen in row 3, columns 2-4; row 4, columns 5-7; row 5, columns 8-10; and row 6, columns 11-13), the EER for all three CMs is either zero or close to zero. This outcome is expected since the CMs are trained using spoofed utterances containing identical artefacts to those in the test data (i.e., utterances generated with the same algorithm and configuration).

However, EERs are notably higher under mismatched conditions. Given that the set of bona fide utterances is constant across all conditions, the variations in EER can be attributed to differences in the spoofed utterances. These results suggest that the artefacts associated with VITS-generated utterances change depending on the training data. We will now delve into a more detailed examination of these findings for each CM.

AASIST – Under mismatched conditions, EERs for AASIST increase but remain reasonably low, with some still below 1%. Notably, EERs continue to be low even when synthetic data is generated using the same model but under varied generation conditions (e.g., V1 & V1.2 - V1.5).

RawNet2 – EERs for RawNet2 are generally much higher under mismatched conditions, and they vary significantly across different generation conditions. For instance, with the V2 training condition (column 6), the EER for the V1 test set is 13.27%, but it drops to 6.63% for the V1.5 test set. This pattern of substantial variation in EERs across different generation conditions is also observed for the V3 and V4 training conditions. Contrasting with AASIST, RawNet2 exhibits difficulties in generalising to different generation conditions.

SSL-AASIST – While EERs for SSL-AASIST remain relatively low, this system shows less robustness compared to AASIST when faced with variability in generation conditions. For example, the EER for the V2 training set jumps from 0.04% for the V1 test set to 0.83% for the V1.5 test set—a nearly 20-fold increase. Despite maintaining a low EER, this result is somewhat unexpected, considering that the SSL front-end is designed to extract high-level representations that typically generalise well across various conditions.

The results discussed above indicate that an adversary can potentially bypass

Table 6.3: Performance in terms of the EER (%) for CMs trained on combined sets V2-V4 and tested against unseen V1 and V1.2-V1.5 attacks.

	Trained on V2-4		
Tested on	AASIST	RawNet2	SSL-AASIST
V1	0	2.20	0
V2	0	2.93	0
V3	0	0.47	0
V4	0	1.37	0
V1.2	0	1.90	0
V1.3	0	0.77	0.03
V1.4	0	2.83	0
V1.5	0	0.87	0.03
Pooled	0	1.79	0.01

a spoofing countermeasure by implementing subtle modifications to the algorithm used for generating spoofed utterances. This tendency is particularly noticeable with the RawNet2 CM, and to a lesser extent, with the AASIST and SSL-AASIST CMs. Consequently, we explore whether CM robustness can be enhanced by training CMs with spoofed utterances generated by multiple attack algorithms with different configurations. To test this hypothesis, we trained a CM using data generated with the V2, V3, and V4 configurations (having fixed noise standard deviation) and tested it with utterances generated using the V1-V1.5 configurations. The results of this experiment are compiled in Table 6.3.

For the AASIST and SSL-AASIST CMs, training with the V2, V3, and V4 datasets resulted in zero or nearly zero EERs for all V1-V1.5 test sets. Comparing the results in Tables 6.2 and 6.3, we observe that training with attacks generated using multiple, differently configured algorithms improves CM generalisation.

This approach also proved beneficial for the RawNet2 CM. The pooled EER of 1.79% for training with V2, V3, and V4 sets is significantly lower than the pooled EERs for RawNet2 presented in Table 6.2. However, there is still considerable variation in EER, indicating that RawNet2 remains vulnerable to certain attack configurations, making it less effective compared to the AASIST and SSL-AASIST

alternatives.

6.4 Summary

Our findings demonstrate that a CM trained with spoofed data from a single attack configuration may be vulnerable to utterances generated using the same algorithm but with different settings. We also find that CM generalisation can be improved by training with spoofed utterances from multiple, varied attack configurations. This approach, a form of data augmentation, is known to be beneficial in related fields like speaker recognition and spoofing detection. Traditional data augmentation methods, such as introducing additive and convolutive noise, enhance CM reliability under varying acoustic and channel conditions. In contrast, spoofing attack augmentation specifically improves generalisation in response to variations in spoofing attacks, even those involving modest changes to an attack algorithm that might otherwise significantly degrade detection performance.

Since most current anti-spoofing databases are created without incorporating spoofing attack augmentation, thoroughly evaluating the impact of such augmentation and training more generalised CMs capable of handling both known and unknown attacks presents a significant challenge. In the next chapter, we propose a framework to support the protection offered by CMs developed using current anti-spoofing databases. Our approach involves supplementing existing CMs with speaker verification systems, to enhance their spoofing detection capabilities.

Chapter 7

Improving generalisation by combining spoofing countermeasure with automatic speaker verification system

The work presented in this chapter is concerned with improving the performance of voice biometric authentication systems by integrating automatic speaker verification (ASV) and spoofing countermeasure (CM) sub-systems into a single, more reliable system. The goal remains the discrimination of bona fide, target trials from anything else, e.g. non-target trials or artificially generated utterances design to manipulate usual system behaviour.

The ASV sub-system [7, 132] is designed to capture speaker characteristics in high-level, compact deep representations suited to the modelling of speaker (dis)similarity. Significant improvements in speaker modelling and verification have been made in recent years [133, 134]. Similar progress in text-to-speech synthesis (TTS) [124, 135] and voice conversion (VC) [136, 137], particularly that stemming from developments in deep learning, nowadays pose a very real threat to ASV reliability. To protect against this threat, auxiliary CM sub-systems [57,60] are used to detect and prevent spoofed or deepfake attacks. CMs capture different characteristics than ASV systems, namely telltale artefacts which serve to identify synthesised, converted or otherwise manipulated utterances.

Given their different tasks, ASV and CM sub-systems, both usually binary classifiers, are typically learned using different purpose-collected databases. Those

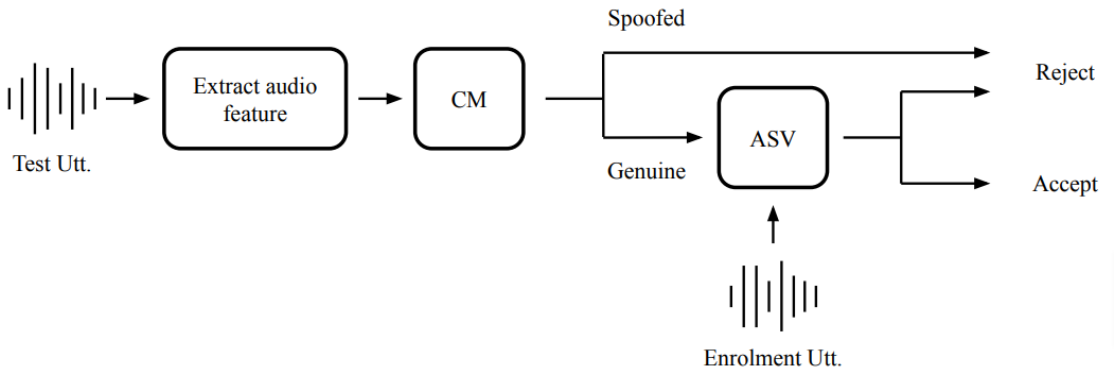


Figure 7.1: Illustration of spoofing CM working in a cascade approach with an ASV system

used for the development of ASV sub-systems [31–33, 138] are huge sets of speech utterances collected from a substantial number of different speakers. In contrast, those used for the development of CMs [15, 36, 139] contain sets of bona fide utterances as well as spoofed utterances generated with a number of different spoofing attack algorithms. These datasets also typically contain data collected from a number of different speakers. Having been learned and optimised, ASV and CM sub-systems are then combined according to some specific integration architecture, an example of which is shown in Figure 7.1. While other integration architectures are also possible [17, 140], the CM is here used as a gate to the ASV sub-system, acting to filter out utterances it classifies as spoofed [16]. The ASV sub-system, operating only upon utterances classified by the CM as being bona fide, then determines whether or not a given test utterance contains the same voice as in the given enrolment utterance. Only utterances classified as bona fide and as containing the same voice as the enrolment utterance are accepted. Anything else is rejected.

Even though ASV and CM sub-systems are designed to solve different, specific problems, their functionalities can (and typically do) overlap – while CMs can help to reject spoofed utterances, they also have the potential to falsely reject bona fide utterances. On the other hand, the ASV sub-system has the potential to reject spoofed utterances, e.g. if they do not match sufficiently well the characteristics of the voice contained in the enrolment utterance. Contrary to common assumptions [92], the two systems are thus not *fully* independent; they function *together* as an integrated solution to the spoofing-robust ASV problem. It is hence

of interest that they be jointly optimised.

We develop further in Section 7.1 the motivation for our work in joint optimisation and present a brief overview of the Spoofing-Aware Speaker Verification (SASV) challenge, the bench-marking framework we use for evaluation. Our approach to joint optimisation is described in Section 7.2. Our experimental setup is described in Section 7.3 with results and analysis being presented in Section 7.4. Conclusions are presented in Section 7.5.

The work presented in this chapter was published in:

Wanying Ge*, Hemlata Tak*, Massimiliano Todisco and Nicholas Evans, "**On The Potential of Jointly-Optimised Solutions to Spoofing Attack Detection and Automatic Speaker Verification**," in *Proc. IberSPEECH 2022* (* equal contribution).

Wanying Ge, Hemlata Tak, Massimiliano Todisco and Nicholas Evans, "**Can Spoofing Countermeasure and Speaker Verification Systems Be Jointly Optimised?**" in *Proc. ICASSP 2023*.

7.1 Spoofing-aware speaker verification challenge

Our work on the joint optimisation of ASV and CM sub-systems is aligned with the objectives of the first Spoofing-Aware Speaker Verification (SASV) challenge [141]. Whereas the well-known ASVspoof challenge supports the design of separate ASV and CM sub-systems, the SASV challenge demands the computation of a single score which encompasses the functionalities of both. In the following, we outline the tasks of speaker verification and spoofing detection and then the spoofing-aware speaker verification task and challenge.

ASV systems operate using speaker embeddings. Utterances are first processed using a deep neural network (DNN) or a convolutive neural network (CNN) to obtain frame-level features. Various pooling layers can be used to aggregate frame-level features into utterance-level features by capturing the temporal dynamics of the input signal [7]. Utterance-level features are then mapped into a new space in which informative speaker embeddings are extracted. Embeddings extracted from utterances produced by the same speaker should have higher similarity than embeddings extracted from utterances produced by different speakers. Various

Table 7.1: Trial types used for performance measurement for three tasks. "+" indicates the positive class and "-" indicates the negative class.

	Bona fide target speaker	Bona fide non-target speaker	Spoofed target speaker
Speaker Verification	+	-	
Spoofing Detection	+		-
Spoofing-aware Speaker Verification	+	-	-

different loss functions [142] can be used in order to minimise the distance between embeddings corresponding to the same speaker, while maximising the distance between embeddings corresponding to different speakers.

Nowadays, almost all CMs are deep-learning-based. CM models are usually less complex (fewer trainable parameters) than ASV models and are almost always binary, two-class classifiers. Given the risk of over-fitting to specific spoofing attacks seen in the training data, so-called one-class solutions [57,143] which model only the bona fide class have also been explored. These systems aim to determine whether an input utterance is *sufficiently* bona fide. More traditional two-class classifiers are still dominant.

Three different trial classes are used for performance assessment. These are listed in the first row of Table 7.1. The ASV sub-system should discriminate bona fide target utterances from bona fide non-target utterances. The CM sub-system is tasked with discriminating between bona fide target utterances and spoofed target utterances. The ASV and CM sub-systems are then combined so that only bona fide target utterances are accepted. Everything else should be rejected.

One objective of the SASV challenge [141] was to study the potential benefit of jointly-optimised ASV and CM combination strategies. However, none of the 23 submissions to the challenge actually explored joint optimisation. All the top-performing systems used either score or embedding level fusion strategies whereby, in contrast to the cascade approach illustrated in Figure 7.1, decisions are made either by combining the scores produced by each sub-system or by learning scores from combined ASV and CM embeddings. More integrated solutions (but not jointly optimised) have also been reported, e.g. [144–147]. These approaches employ a single model to capture both speaker characteristics and spoofing artefacts.

The performance of these approaches is inferior to that of fusion-based alternatives [47–51, 148]. The objectives of our work reported in this chapter are to determine whether joint optimisation can help to narrow the performance gap and to determine whether jointly-optimised systems might one day even outperform competing approaches.

7.2 The proposed optimisation framework

We present in this section our integrated, jointly-optimised SASV system. It contains three components, namely an ASV sub-system, a CM sub-system and a back-end classifier. They are combined to produce a single output score. Being jointly-optimised, both ASV and CM model parameters are updated according to a single loss function. The optimisation criteria, however, is also a design choice. This implies that we can fix the sub-system model parameters and then update only those of the back-end classifiers, or update all model parameters simultaneously. This flexibility allows for a meaningful comparison of sub-system behaviour both before and after joint optimisation, as well as the assessment of joint optimisation itself upon the performance of the system as a whole.

7.2.1 ASV sub-system

We use the model described in [142] as the ASV sub-system, namely a ResNet34 model with squeeze-and-excitation (SE) blocks [149] (ResNetSE34). The first feature extraction layer serves to decompose raw input speech waveforms into spectro-temporal representations, more specifically, log-scale Mel-filterbank features. Four convolutional layers with SE blocks are then used to process the input features and extract a deep, compact representation. Attentive statistics pooling (ASP) [150] is used to aggregate and project frame-level features into a fixed-length, utterance-level embedding. The network is optimised using a combination of softmax and angular prototypical loss [151]. The ASV sub-system is used to extract embeddings from both enrolment and test utterances giving $e_{\text{enr}}^{\text{ASV}}$ and $e_{\text{tst}}^{\text{ASV}}$ embeddings respectively.

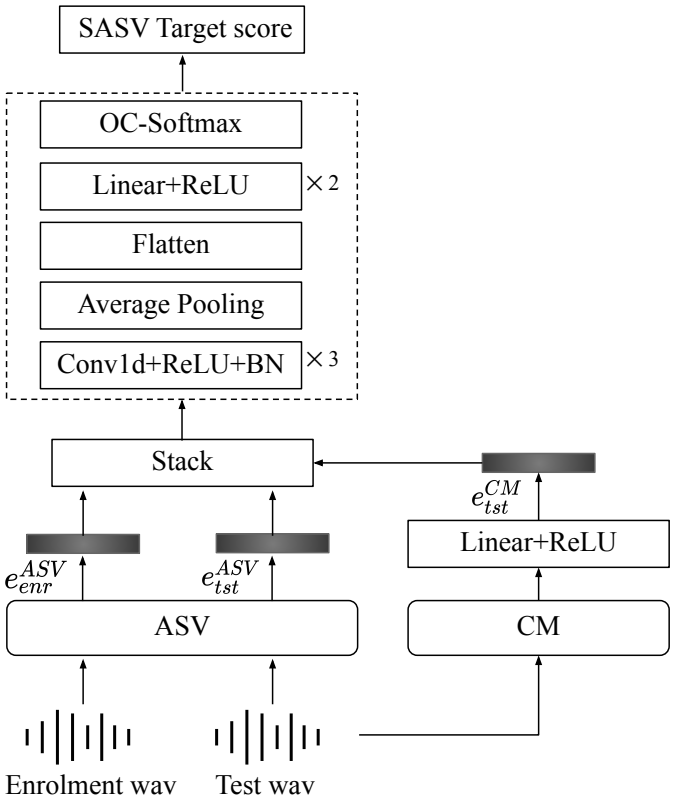


Figure 7.2: Framework for separate and joint optimisation. The ASV sub-system extracts speaker embeddings from both enrolment and test utterances, while the CM sub-system only extracts one embedding from the test utterance. This CM embedding is then projected to the same dimension as the ASV embeddings through a linear layer. Finally, the three embeddings are stacked along a new dimension and processed by a CNN-based back-end to calculate the final SASV target score.

7.2.2 CM sub-system

We use the AASIST model described in [78] as the CM sub-system. It is an end-to-end (E2E) system designed to ingest raw input speech waveforms. A RawNet2 based encoder [60] is used with graph attention networks [61] to decompose speech waveforms into high-dimensional temporal and spectral representations which are subsequently integrated into a single, combined representation. A readout operation is used to extract the CM embedding from the integrated representation. The embeddings produced by the AASIST model are then further processed through a linear layer in order to extract a CM embedding $e_{\text{tst}}^{\text{CM}}$ which has the same di-

Table 7.2: Details of the ASVspoof 2019 LA and FAD training partitions used for all experiments reported in this paper.

Database	# of speakers	# of bona fide	# of spoofed	# of attacks
ASVspoof train	20	2580	22800	4 TTS, 2 VC
FAD train	40	3200	25600	8 TTS

mensionality as the ASV embeddings. The network is optimised using a weighted cross-entropy loss function.

7.2.3 Backend classifier

Embeddings extracted from both sub-systems are stacked to enable a convolutional neural network (CNN) to capture both variances stemming from test and enrolment ASV embeddings and the CM embedding. A 1-dimensional adaptive average pooling layer is used to aggregate the set of embeddings and an OC-Softmax layer [57] is used to generate the final score, an indication of support for the bona fide target class.

7.3 Experiment setup

We describe in this section the databases, protocols and evaluation metrics used for our work in addition to implementation details.

7.3.1 Database

The ResNetSE34 ASV sub-system is pre-trained using the development set of the VoxCeleb2 database [32]. The MUSAN corpus [39] and simulated room impulse response (RIR) filters [38] are used for data augmentation (DA). The VoxCeleb1 [31] test set is used to select the best model. The AASIST CM sub-system is trained using the training partition of the ASVspoof 2019 logical access (LA) database [15], without DA. The development set of the same database is used to select the best model. The SASV system is trained using the training partition of the ASVspoof 2019 LA database (same data as for CM training, also without DA).

Also reported below are a set of experiments in which we use an pool of training data which is augmented using utterances sourced from the Fake Audio Detection (FAD) database [35]. Details of the ASVspoof and FAD training partitions are

Table 7.3: SASV utterance types and the corresponding proportions of test utterances for fixed and joint optimisation.

	Utterance type	Prop. in Fixed	Prop. in Joint
①	Bona fide, Target spk	50%	25%
②	Bona fide, Non-target spk	25%	25%
③	Spoofed, Target spk	25%	25%
④	Spoofed, Non-target spk	None	25%

shown in Table 7.2. Like the ASVspoof 2019 LA database, the FAD database contains bona fide utterances collected from a number of different speakers and spoofed utterances generated using a number of different algorithms. A key difference is that the ASVspoof database contains utterances in English, while the FAD database contains utterances in Mandarin. As shown in Table 7.2, the training partition of the FAD database contains a similar number of bona fide and spoofed utterances. However, the number of speakers is twice that for the ASVspoof database. This auxiliary data is used for training only. The development and evaluation sets are not changed.

7.3.2 Protocols

Whereas the usual ASVspoof protocol is used for CM training, the SASV system is trained using a specific protocol comprising three different trial classes, each comprising a pair of utterance types. These are illustrated in Table 7.3. The enrolment utterance is always a bona fide utterance from the target speaker, labelled ①. The test utterance can be any one of four different utterance types. The first three, labelled ①, ② and ③, correspond to the three trial types shown in Table 7.1. An SASV system should only produce an accept decision when the test utterance is a bona fide target utterance ①, and produce a reject decision when the test utterance is either a bona fide non-target utterance ② or a spoofed target utterance ③.

The last two columns in Table 7.3 indicate the proportions of each test utterance type used for system training. Proportions shown in column 3 are the same as for the SASV challenge protocol, where 50% of the training data is bona fide target test utterances ①, while the remaining 50% is split equally between bona

fide non-target ② and spoofed target ③ test utterances. The remaining type of spoofed non-target utterances ④ is never used for training; by definition, spoofs are always *targeted*. Nonetheless, we found that the use of type ④ utterances during training (last row in Table 7.3) is beneficial in the case of joint optimisation (but not for fixed systems). Thus, for joint optimisation experiments, we use all four types of utterances, and in equal proportion, as shown in the last column of Table 7.3.

In the case of SASV, the negative class is hence the union of utterance types ②, ③ and ④. For evaluation of the CM, the task of which is to discriminate between spoofed and bona fide utterances, utterance type ② is within the positive class. During joint optimisation, this conflict between SASV and CM classes has potential to degrade CM and/or SASV performance. We use utterance type ④ as a means to balance this conflict; type ④ utterances belong to the negative class for both SASV and CM tasks. This may also explain why, as we shall later, the use of type ④ utterances is not beneficial in the case of fixed training for which there is no such conflict.

7.3.3 Metrics

We use estimates of the equal error rate (EER) to evaluate performance. The so-called SASV-EER [141] is used to assess performance when the positive class comprises only bona fide target trials ①, while the negative class is a union of both bona fide non-target trials ② and spoofed target trials ③. Spoofed non-target trials ④ are used only for training and not for testing, hence they play no role in computation of SASV-EER results. The speaker verification EER (SV-EER) and the spoofing detection EER (SPF-EER) are also used to provide further insights into system behaviour and to show performance in the case of specific ASV and CM sub-tasks. While the positive class for the computation of both the SV-EER and SPF-EER comprises the same bona fide targets trials ① as for the SASV-EER, there are differences in the negative class. For the SV-EER, it comprises only bona fide non-target trials ② whereas, for the SPF-EER, it comprises only spoofed target trials ③.

7.3.4 Implementation details

We used default settings for the pre-training of ASV [142] and CM [78] sub-systems. The ASV sub-system uses 64-dimensional log-scale Mel-filterbank features which are extracted using a 25 ms Hamming window with a 10 ms shift per frame. Speaker embeddings are of dimension 512. The CM sub-system produces embeddings of dimension 160. Readers are referred to [142] and [78] for further details.

For SASV training, the CM embedding (produced by the AASIST model) is first transformed to 512 dimensions (using the linear+ReLU layer) and then stacked with the two speaker embeddings. The resulting, stacked embedding is processed using three 1D convolutional layers and an average pooling layer. Scores are computed from the resulting deep representation using an OC-Softmax layer [57]. The scale factor of the OC-Softmax layer is set to 10. The margin for the positive class is set to 0.8, while that for the negative class is set to 0.2.

SASV model parameters are updated for 20 epochs. The initial learning rate is set to 5e-5, with a decay factor of 0.95 for every 200 batches. The batch size is set to 20. We select the best model according to the SASV-EER for the development set. The final SASV-EER for the evaluation set is an average of 5 independent runs, each with a different random seed. All experimental work was performed using a single NVIDIA GeForce RTX 3090 GPU.

7.4 Results

We present results for the proposed system with both fixed and jointly-optimised training regimes, as well as for SASV challenge baselines. We then present analyses which aim to explain differences in performance and the benefit of joint optimisation.

7.4.1 Comparison of results for fixed and joint optimisation

A comparison of results for the proposed framework under fixed and joint optimisation is presented in Table 7.4. EERs are reported both for the SASV 2022 development (`dev`) and evaluation (`eval`) partitions. Also shown are results for the two SASV challenge baselines. In terms of SASV-EER results, the proposed system outperforms the two challenge baselines under both training regimes. Except for our jointly-optimised solution, all systems use fixed ASV and CM sub-systems

Table 7.4: Results for pre-trained, jointly-optimised and baseline systems for SASV 2022 development and evaluation partitions.

System	SASV-EER		SV-EER		SPF-EER	
	dev	eval	dev	eval	dev	eval
Pre-trained, fixed	0.73	1.15	1.41	1.27	0.48	1.08
Joint optimisation	1.15	1.49	2.34	2.34	0.31	0.80
Baseline1-v2 [141]	1.01	1.71	1.99	1.66	0.23	1.76
Baseline2 [152]	4.85	6.37	12.87	11.48	0.13	0.78

for the extraction of embeddings, with only the back-end classifier being further optimised.

Results for the pre-trained, fixed configuration give the lowest SV-EER for both `dev` and `eval` sets. While SPF-EERs are relatively higher than for the best baseline, the overall SASV-EERs for both sets are the lowest among the four systems. Results shown in Table 7.4 do not support our initial hypothesis that joint optimisation is beneficial. SASV-EERs for the pre-trained, fixed system (0.73% for `dev` and 1.15% for `eval`) are noticeably better than those for joint optimisation (1.15% for `dev` and 1.49% for `eval`). The jointly-optimised system achieves a lower SPF-EER for both the development (0.31% compared to 0.48%) and evaluation (0.80% compared to 1.08%) sets, meanwhile SV-EERs are higher (2.34% compared to 1.41% for `dev` and 2.34% compared to 1.27% for `eval`). Joint optimisation results in better spoofing detection but worse speaker verification, and it is the degradation to the latter which then leads to worse SASV-EER performance as well.

7.4.2 Analysis on speaker verification performance

Results presented in Table 7.4 show that the bottleneck for the jointly-optimised SASV system is poor speaker verification performance (SV-EER). We also notice that, except for the jointly-optimised system, SV-EERs are higher for the development set than for the evaluation set. The same trend can also be observed with many of the SASV challenge submissions [50, 51, 148]. This finding suggests that, in terms of speaker verification, the evaluation set is *easier* than the development

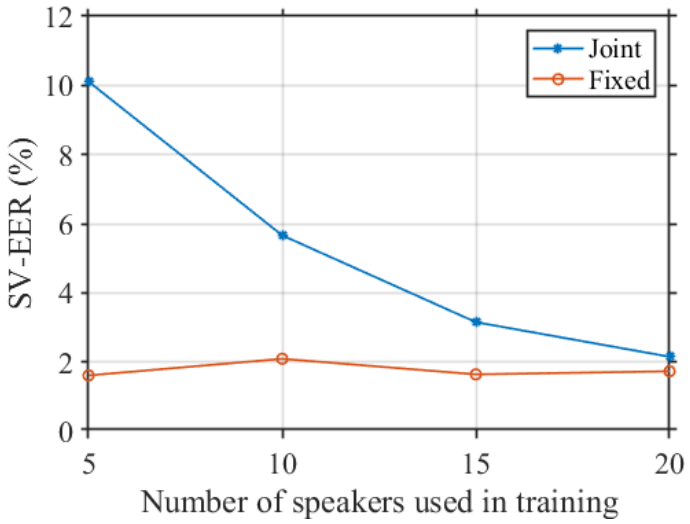


Figure 7.3: SV-EERs estimated using the development partition for pre-trained, fixed and jointly-optimised systems as a function of the number of speakers in the training partition.

set. The different trend observed for the jointly-optimised system, indicates that it over-fits to the speakers in the development set.

The problem of over-fitting is the result of insufficient diversity in the training data. In our case, this translates to a lack of *speaker* diversity. Although the VoxCeleb2 [32] database used for the pre-training of ASV sub-systems contains data collected from over 5000 speakers, the SASV 2022 training data which is used for fine-tuning, contains data collected from only 20 speakers. This number is likely far too small and a reason for the degradation in SV-EER observed for the jointly-optimised system.

To test this hypothesis, we designed an experiment to observe the change in performance when using training data collected from a lower number of speakers; we cannot use a greater number, at least not using data sourced from the same database. We re-trained the system, again under both fixed and joint optimisation, but this time with data from a different number of speakers. Training is conducted using the same set of trials as before, namely the set of ①, ② and ③ trials. SV-EER results for the development set are shown in Figure 7.3. The SV-EER for the fixed system is stable, at around 2%. This finding shows that the system has reasonable performance after pre-training and that fine-tuning of the back-

Table 7.5: Averaged results for pre-trained, jointly-optimised systems for SASV 2022 evaluation partitions.

Training data	Configuration	SASV-EER	SV-EER	SPF-EER
ASVspoof	Fixed	1.15	1.27	1.08
	Joint	1.49	2.34	0.80
ASVspoof +	Fixed	1.52	1.85	1.27
FAD	Joint	1.74	2.66	1.09
ASVspoof +	Fixed	1.72	1.47	1.84
FAD bona fide only	Joint	1.26	1.77	0.83

end classifier has little impact. However, SV-EER results for the jointly-optimised system increase substantially as the number of speakers is reduced. This finding supports our suspicion that joint optimisation results in over-fitting to the speakers in the training data.

7.4.3 Results with external speakers

Given that results degrade when the number of speakers is *reduced*, it is of interest to find a solution to *increase* their number. This cannot be achieved using data from the same database since it is already exhausted. Accordingly, we augment the pool of training data using utterances sourced from the FAD database [35], details of which are shown in Table 7.2.

Results for the evaluation set are shown in Table 7.5. Results in the second and third rows are the same as those for jointly-optimised and fixed systems in Table 7.4. Rows four and five show results when using both ASVspoof and FAD training data. Results do not improve, neither for fixed or jointly-optimised systems; SASV-EERs increase from 1.15% (fixed) and 1.49% (joint) to 1.52% (fixed) and 1.74% (joint). In addition, results for the jointly-optimised system are still worse than those for the fixed system. We observe the same trend for the SV-EER and SPF-EER. Worse speaker verification and spoofing detection performance can be caused by different reasons. First, the language mismatch might complicate the use of information learned from Mandarin-language data; SV-EERs degrade when FAD data is used for training (1.27% to 1.85% for fixed; 2.34% to 2.66% for joint).

Similarly, SPF-EERs are also worse (1.08% to 1.27% for fixed; 0.80% to 1.09% for joint). This might be because of the differences in spoofing attacks in the ASVspoof and FAD training data which complicate the use of information learned from the latter.

While the language mismatch cannot be avoided, we can exclude the spoofed utterances among the FAD training data in order to reduce the influence from mismatch in the spoofing attacks. Results for this set up are shown in the last two rows of Table 7.5. Performance is still worse that that using only ASVspoof training data. These results nonetheless show some promise. The SASV-EER for the jointly-optimised system is now better than that for the fixed system (1.26% compared to 1.72%). This result is not far away from the best (1.15%).

The improvement in SASV performance for the jointly-optimised system, even if still not the best, stems from the improvement in speaker verification performance. The SV-EER of 1.77% is the lowest of all results for jointly-optimised systems, even if it is still higher than that for the fixed system under identical training conditions. SPF-EER results also improve when spoofed data is excluded (1.09% for ASVspoof + FAD compared to 0.83% for ASVspoof + FAD bona fide only). These results confirm that additional, external data collected from other speakers is beneficial to joint optimisation.

7.4.4 Analysis on sub-system complementarity

Results discussed above indicate that joint optimisation remains inferior to a pre-trained competitor, even if the use of additional external data reduces the performance gap. We have so far learned that speaker verification performance is the bottleneck. Spoofing detection performance actually improves under joint optimisation. This finding is not necessarily surprising since, as argued above, the ASV sub-system also has potential for spoofing detection when speaker characteristics are not well reflected. Improvements to spoofing detection are a sign that the ASV sub-system is complementary to the CM sub-system. On other other hand, the CM has no access to the enrolment utterance, thus degradation to speaker verification performance likely come solely from the ASV sub-system. This argument, though, does not explain why SV-EER and SPF-EER results for both fixed and jointly-optimised systems both degrade when the pool of training data

Table 7.6: Averaged results for pre-trained, jointly-optimised sub-systems for SASV 2022 evaluation partitions. Results under fixed configuration are same since the networks are identical. Results in **boldface** indicate better performance for jointly-optimised systems than for corresponding fixed, independently-optimised systems.

Training data	Configuration	ASV sub-system			CM sub-system		
		SASV-EER	SV-EER	SPF-EER	SASV-EER	SV-EER	SPF-EER
ASVspoof	Fixed	19.70	1.27	25.75	24.50	49.01	0.65
	Joint	**13.47**	1.84	**17.43**	**23.65**	**46.80**	1.02
ASVspoof +	Fixed	19.70	1.27	25.75	24.50	49.01	0.65
FAD	Joint	**8.57**	1.73	**10.82**	**22.83**	**46.37**	1.81
ASVspoof +	Fixed	19.70	1.27	25.75	24.50	49.01	0.65
FAD bona fide only	Joint	**8.58**	1.49	**10.60**	24.66	**47.99**	1.35

is augmented with FAD data.

The higher SPF-EER can be caused by either ASV or CM sub-systems. We designed another set of experiment to examine the complementarity between the two. All results presented in Table 7.6 are for the same systems for which results are reported in Table 7.5, though for separate sub-systems instead of the full SASV system. When the jointly-optimised system is trained using only the ASVspoof database, we observe an improvement in SASV-EER for both sub-systems. Though the ASV sub-system, when jointly optimised, has a higher SV-EER than the fixed system, the SPF-EER is lower; this accounts for the lower SASV-EER. The ASV sub-system hence learns some capability to distinguish between bona fide and spoofed trials. In so doing, it sacrifices some speaker verification capability. On the other hand, the spoofing detection performance of the the CM sub-system degrades, but is compensated somehow by it learning some speaker verification capability, albeit very low.

The SASV-EER for the full system (Table 7.5) still degrades. The lower SPF-EER for the ASV sub-system, but the higher SPF-EER for the CM sub-system, results in a lower SPF-EER for the full system. The higher SV-EER for the ASV sub-system and the lower SV-EER for the CM sub-system result in a higher SV-EER for the full system. Our explanation is that, though the SV-EER for CM sub-system decreases, it is not beneficial since the CM has no access to the enrol-

ment utterance – the CM normally hence has no capacity for speaker verification and should have an SV-EER of around 50%. The lower SV-EER for the CM is instead a sign of speaker awareness, specifically that it performs differently for different speakers. Such speaker-related information is not beneficial to the speaker verification task, and is why the SV-EER as well as the SASV-EER for the full system is worse.

The remainder of results in Table 7.6 confirm that spoofing attacks contained in the FAD database are not beneficial to the detection of attacks contained in the ASVspoof database. The SPF-EER for the CM sub-system increases (from 1.02% to 1.81%) when the pool of training data contains both bona fide and spoofed utterances from the FAD database. When spoofed utterances are excluded, it decreases (from 1.81% to 1.35%). Meanwhile, the SPF-EER of the ASV sub-system is relatively stable (10.82% to 10.60%). When jointly-optimised, the use of data corresponding to additional speakers can help reduce the speaker over-fitting of both sub-systems, with the SV-EER of the ASV sub-system decreasing (from 1.73% to 1.49%), and that of the CM sub-system *increasing* (46.37% to 47.99% – it should be 50%).

7.5 Summary

In this chapter, we present our work to investigate the merit of jointly-optimised solutions to combine automatic speaker verification (ASV) and spoofing countermeasure (CM) sub-systems as a solution to the spoofing-aware speaker verification task. The joint optimisation of each sub-system can exploit complementary information and has the potential to achieve better performance than when the two systems are optimised independently. Despite the benefit of doing so, we identified challenges and a key performance bottleneck – the lack of speaker diversity in training data. Experiments confirm that the addition of external speaker data helps to reduce speaker over-fitting and to improve performance. A detailed analysis of ASV and CM sub-system complementarity provides additional evidence of sub-system synergy in spoofing detection, while improvements to speaker verification stem solely from the ASV sub-system. Future work includes the investigation of different training strategies to better address over-fitting. Other directions include the design of more efficient models to better exploit sub-system complementarity.

Chapter 8

Conclusions and future directions

In this chapter, we present the summary of the work presented in this thesis. We first present the key the contributions and findings in Section 8.1. Potential directions for future work are presented in Section 8.2.

8.1 Key contributions

The central topic of this thesis is the exploration of Automatic Speaker Verification (ASV) system vulnerabilities to spoofing attacks. We aim to enhance the reliability of voice anti-spoofing systems and their integration with ASV systems. This thesis not only advances the field of voice anti-spoofing but also lays the groundwork for future research on secure, interpretable, and integrated spoofing resistant ASV systems. Thesis contributions are summarised in the following. Chapters 3 and 4 focus on the automated design of countermeasure (CM) systems. Chapter 5 provides insights into the system decision-making process. Chapter 6 demonstrates that variability in spoofing model training can undermine detection and Chapter 7 reports the development of joint optimisation of ASV and CM to overcome such vulnerabilities and to improve reliability.

In **Chapter 3**, we present the first application of differentiable architecture search (DARTS), specifically Partially Connected DARTS (PC-DARTS), to spoofing detection. This novel approach enables the automated learning of deep neural network architectures, and their optimisation through backpropagation and hardware acceleration. The models generated via PC-DARTS exhibit competitive performance, as shown in Table 8.1. Our best model achieves a min-tDCF of 0.09 for

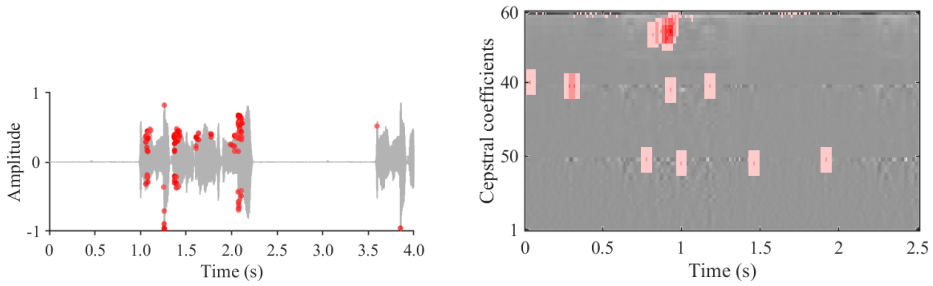
Table 8.1: Performance comparison between PC-DARTS and Raw PC-DARTS models and competing state-of-the-art systems reported in the literature on ASVspoof LA evaluation partition. Result reproduced from Table 3.3 and 4.3.

Systems	**Features**	**min-tDCF**	**EER**	**Params**
Res-TSSDNet [21]	waveform	0.0482	1.64	0.35M
Raw PC-DARTS Mel-F	waveform	0.0517	1.77	24.48M
Raw PC-DARTS Linear-L	waveform	0.0583	2.10	24.40M
Res2Net [95]	CQT	0.0743	2.50	0.96M
PC-DARTS $(16, 64)$	LFCC	0.0914	4.96	7.51M
PC-DARTS $(4, 16)$	LFCC	0.0992	5.53	0.14M
Challenge Baseline-1	LFCC	0.2116	8.09	-
Challenge Baseline-2	CQCC	0.2366	9.57	-

the ASVspoof 2019 database, closely following a top-performing Res2Net system. Our second-best model, with a min-tDCF of 0.1, requires 85% fewer parameters than the leading Res2Net competition, highlighting the efficiency and potential of PC-DARTS for evolving network architectures with reduced computational complexity and minimal manual intervention. This breakthrough underscores the feasibility and benefits of employing PC-DARTS for spoofing detection, encouraging further exploration in this direction.

In **Chapter 4** we introduce Raw PC-DARTS, an end-to-end differentiable architecture search methodology which enables the comprehensive automation of learning pre-processing operations, network architectures, and parameters directly from raw waveform inputs. Such an approach ensures that the entire model is optimised in unison, leading to a highly efficient and competitive system. Although the best performing Raw PC-DARTS system is still achieved with a fixed Mel-scale front-end (1.77% as shown in Table 8.1), performance of the system with jointly searched front-end and architecture is only marginally behind (2.10%). The performance of our models ranks among the top single-system results at the time of publication.

In **Chapter 5** we introduce the use of SHapley Additive exPlanations (SHAP) to shed light on the variation in model behaviour and the distinct artefacts left



(a) Raw PC-DARTS, LA_T_1859200 (b) PC-DARTS, LA_T_1859200

Figure 8.1: SHAP values for Raw PC-DARTS (time-domain waveform) and PC-DARTS (Linear-frequency cepstrum coefficient feature) for utterance LA_T_1859200 - '*We will do so again*'. Figure reproduced from Figure 5.3.

Table 8.2: CM performance in terms of the EER (%) in two training conditions. The first training set only contains spoofed V1 set, the second contains V2-4 sets. CMs are tested on both V1-4 and V1.2-V1.5. Results reproduced from Table 6.2 and 6.3.

	Trained on V1			**Trained on V2-4**		
Tested on	AASIST	RawNet2	SSL-AASIST	AASIST	RawNet2	SSL-AASIST
V1	0	0	0	0	2.20	0
V2	0.50	6.27	0.07	0	2.93	0
V3	2.43	8.50	0.03	0	0.47	0
V4	1.20	7.93	0	0	1.37	0
V1.2	0	0.67	0	0	1.90	0
V1.3	0	0.03	0	0	0.77	0.03
V1.4	0	0.93	0	0	2.83	0
V1.5	0	0.10	0	0	0.87	0.03

behind by different spoofing attacks. Through SHAP analysis, we discovered notable differences in how our models react to spoofed utterances generated with different algorithms. Specifically, our PC-DARTS model tends to focus on non-speech intervals for detection, whereas the Raw PC-DARTS model relies more on speech intervals, as shown in Figure 8.1. This unexpected finding helps explain the performance variation between the two models and underscores the importance of considering both speech and non-speech intervals for spoofing detection.

Table 8.3: SASV performance results for pre-trained, jointly-optimised systems trained with original ASVspoof data and with additional FAD bona fide data. Result reproduced from Table 7.5.

Training data	Configuration	SASV-EER	SV-EER	SPF-EER
ASVspoof	Fixed, full	1.15	1.27	1.08
	- ASV	19.70	1.27	25.75
	- CM	24.50	49.01	0.65
	Joint, full	1.49	2.34	0.80
	- ASV	13.47	1.84	17.43
	- CM	23.65	46.80	1.02
ASVspoof + FAD bona fide only	Fixed, full	1.72	1.47	1.84
	- ASV	19.70	1.27	25.75
	- CM	24.50	49.01	0.65
	Joint, full	1.26	1.77	0.83
	- ASV	8.58	1.49	10.60
	- CM	24.66	47.99	1.35

We further explored the variation in spoofing artefacts in **Chapter 6**. Our hypothesis is that the variability observed in deep-learning-based voice spoofing detection models—caused by different initialisation, hyper-parameters, or training data partitions—might also apply to spoofing generation models. To test this hypothesis, we focus on the generalisation of CM systems when trained using data generated with differently-configured algorithms. Our results confirm that CM systems trained on a broader range of spoofed data exhibit improved detection capabilities, underscoring the importance of diversifying training data to enhance the robustness and generalisation of spoofing detection systems against evolving spoofing techniques.

In **Chapter 7** we report our attempts to improve CM robustness and generalisation using jointly-optimised ASV systems. The corresponding results are reproduced in Table 8.3. The integration of ASV and CM sub-systems aims to leverage their complementary strengths. However, we encountered a significant challenge, namely the limited diversity of speakers in the training data, which leads to speaker over-fitting. Our experiments show that the importance of exter-

nal speaker data mitigates this issue, enhancing overall system performance.

8.2 Directions for future research

In the following, we outline potential future directions which provide opportunities to extend the work presented in this thesis:

1. **Exploring a unified ASV and CM system architecture**: Our exploration with PC-DARTS and Raw PC-DARTS focused upon the utilisation of neural architecture search for spoofing detection. Although DARTS has shown promise in ASV tasks [153], the architectures it discovers are specifically tuned for ASV, with no assurance of robustness to spoofing detection. Our investigations into the joint optimisation of ASV and CM systems highlight their complementary nature. It is, therefore, compelling to pursue a unified architecture that could harness this synergy more effectively, aiming to address both tasks simultaneously, and, once again, with minimised human intervention.

2. **Advancing explainability techniques**: While methods like SHAP offer valuable insights into feature importance, there remains a gap in delivering intuitive and meaningful explanations. The General Data Protection Regulation (GDPR) mandates that data subjects have the right to obtain meaningful information about the logic involved in automated decision-making. However, explanations via heat-maps provide limited insight and often demand specialised knowledge for interpretation. Investigating alternative explanatory forms, such as textual explanations, could offer more accessible insights to supplement heat map-based explanations.

3. **Spoofing-oriented voice generation**: The goal of most text-to-speech (TTS) systems is to generate voices that ① are realistic and ② mimic the sound of a target human ③ speaking the given text. While CMs aim to identify ① and ASV systems aim for ②, in automated, text-independent scenarios, the need for ③ (i.e., coherent text without skipping or repeating words) is not paramount. This frees up TTS networks to focus on evading ASV and CM detection. Building on insights from our work in Chapter 6,

future work should examine whether merely sounding realistic is sufficient to elude CMs, or if sounding artificial in novel ways might also be effective.

4. **Audio-visual deepfake generation and detection**: Though the joint optimisation of ASV and CM systems does not contribute technically a multi-modal approach, the two are still complementary to each other. There is still nonetheless an opportunity to improve biometric verification systems using multi-modal cues from both facial images as well as speech recordings. Similar to the ASVspoof databases, the development of a robust, generalised audio-visual deepfake detection system necessitates the creation of high-quality databases.

Bibliography

[1] Z. Li, Z. Zhao, W. Wang, P. Zhang, and Q. Zhao, "Explore long-range context features for speaker verification," *Applied Sciences*, vol. 13, no. 3, p. 1340, 2023.

[2] H. Tak, "End-to-end modeling for speech spoofing and deepfake detection," Ph.D. dissertation, Sorbonne University, 2023.

[3] W. Ge, X. Wang, J. Yamagishi, M. Todisco, and N. Evans, "Spoofing attack augmentation: Can differently-trained attack models improve generalisation?" in *Proc. ICASSP 2024*, 2024, pp. 12 531–12 535.

[4] D. A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," *Speech Communication*, vol. 17, no. 1, pp. 91–108, 1995.

[5] V. Dellwo, M. Huckvale, and M. Ashby, "How is individuality expressed in voice? An introduction to speech production and description for speaker classification," *Speaker Classification I: Fundamentals, Features, and Methods*, pp. 1–20, 2007.

[6] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. INTERSPEECH 2015*, 2015, pp. 3214–3218.

[7] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. INTERSPEECH 2020*, 2020, pp. 3560–3564.

[8] T. Zhou, Y. Zhao, and J. Wu, "Resnext and res2net structures for speaker verification," in *2021 IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 301–307.

[9] J. S. Chung, J. Huh, S. Mun, M. Lee, H. S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, "In defence of metric learning for speaker recognition," in *Proc. INTERSPEECH 2020*, 2020, pp. 2977–2981.

[10] L. Li, R. Nai, and D. Wang, "Real additive margin softmax for speaker verification," in *Proc. ICASSP 2022*, 2022, pp. 7527–7531.

[11] D. A. Reynolds, "An overview of automatic speaker recognition technology," in *Proc. ICASSP 2002*, 2002, pp. 4072–4075.

[12] M. Sahidullah, H. Delgado, M. Todisco, A. Nautsch, X. Wang, T. Kinnunen, N. Evans, J. Yamagishi, and K.-A. Lee, "Introduction to voice presentation attack detection and recent advances," *Handbook of Biometric Anti-Spoofing: Presentation Attack Detection and Vulnerability Assessment*, pp. 339–385, 2023.

[13] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 Challenge: Assessing the limits of replay spoofing attack detection," in *Proc. INTERSPEECH 2017*, 2017, pp. 2–6.

[14] J. Frank and L. Schönherr, "WaveFake: A data set to facilitate audio deepfake detection," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[15] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. H. Kinnunen, and K. A. Lee, "ASVspoof 2019: Future horizons in spoofed and fake audio detection," in *Proc. INTERSPEECH 2019*, 2019, pp. 1008–1012.

[16] M. Sahidullah, H. Delgado, M. Todisco, H. Yu, T. Kinnunen, N. Evans, and Z.-H. Tan, "Integrated spoofing countermeasures and automatic speaker verification: An evaluation on ASVspoof 2015," in *Proc. INTERSPEECH 2016*, 2016, pp. 1700–1704.

[17] M. Todisco, H. Delgado, K. A. Lee, M. Sahidullah, N. Evans, T. Kinnunen, and J. Yamagishi, "Integrated presentation attack detection and automatic speaker verification: Common features and gaussian back-end fusion," in *Proc. INTERSPEECH 2018*, 2018, pp. 77–81.

[18] T. Masuko, T. Hitotsumatsu, K. Tokuda, and T. Kobayashi, "On the security of HMM-based speaker verification systems against imposture using synthetic speech," in *Proc. EUROSPEECH 1999*, 1999, pp. 1223–1226.

[19] H. Yu, Z.-H. Tan, Z. Ma, R. Martin, and J. Guo, "Spoofing detection in automatic speaker verification systems using DNN classifiers and dynamic acoustic features," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4633–4644, 2017.

[20] H. Yu, Z.-H. Tan, Y. Zhang, Z. Ma, and J. Guo, "DNN filter bank cepstral coefficients for spoofing detection," *IEEE Access*, vol. 5, pp. 4779–4787, 2017.

[21] G. Hua, A. B.-j. Teoh, and H. Zhang, "Towards end-to-end synthetic speech detection," *IEEE Signal Processing Letters*, vol. 28, pp. 1265–1269, 2021.

[22] G. Lavrentyeva, S. Novoselov, A. Tseren, M. Volkova, A. Gorlanov, and A. Kozlov, "STC antispoofing systems for the ASVspoof2019 challenge," in *Proc. INTERSPEECH 2019*, 2019, pp. 1033–1037.

[23] P. Nagarsheth, E. Khoury, K. Patil, and M. Garland, "Replay attack detection using DNN for channel discrimination," in *Proc. INTERSPEECH 2017*, 2017, pp. 97–101.

[24] M. Todisco, H. Delgado, and N. Evans, "A new feature for automatic speaker verification anti-spoofing: Constant Q cepstral coefficients," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2016)*, 2016, pp. 283–290.

[25] Z. Lei, Y. Yang, C. Liu, and J. Ye, "Siamese convolutional neural network using gaussian probability feature for spoofing speech detection," in *Proc. INTERSPEECH 2020*, 2020, pp. 1116–1120.

[26] X. Wang and J. Yamagishi, "A comparative study on recent neural spoofing countermeasures for synthetic speech detection," in *Proc. INTERSPEECH 2021*, 2021, pp. 4259–4263.

[27] N. Müller, F. Dieckmann, P. Czempin, R. Canals, K. Böttinger, and J. Williams, "Speech is silver, silence is golden: What do ASVspoof-trained models really learn?" in *Proc. ASVspoof 2021 Workshop*, 2021, pp. 55–60.

[28] B. Chettri, S. Mishra, B. L. Sturm, and E. Benetos, "Analysing the predictions of a cnn-based replay spoofing detection system," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 92–97.

[29] B. Chettri, E. Benetos, and B. Sturm, "Dataset artefacts in anti-spoofing systems: A case study on the ASVspoof 2017 benchmark," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 3018–3028, 2020.

[30] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *International Conference on Learning Representations*, 2020.

[31] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large scale speaker identification dataset," in *Proc. INTERSPEECH 2017*, 2017, pp. 2616–2620.

[32] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *Proc. INTERSPEECH 2018*, 2018, pp. 1086–1090.

[33] G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds, "The NIST speaker recognition evaluation–overview, methodology, systems, results, perspective," *Speech communication*, vol. 31, no. 2-3, pp. 225–254, 2000.

[34] Y. Fan, J. Kang, L. Li, K. Li, H. Chen, S. Cheng, P. Zhang, Z. Zhou, Y. Cai, and D. Wang, "CN-Celeb: A challenging chinese speaker recognition dataset," in *ICASSP 2020*, 2020, pp. 7604–7608.

[35] H. Ma, J. Yi, C. Wang, X. Yan, J. Tao, T. Wang, S. Wang, L. Xu, and R. Fu, "FAD: A Chinese dataset for fake audio detection," *arXiv preprint:2207.12308*, 2022.

[36] J. Yamagishi, X. Wang, M. Todisco, M. Sahidullah, J. Patino, A. Nautsch, X. Liu, K. A. Lee, T. Kinnunen, N. Evans, and H. Delgado, "ASVspoof 2021: Accelerating progress in spoofed and deepfake speech detection," in *Proc. ASVspoof 2021 Workshop*, 2021, pp. 47–54.

[37] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.

[38] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. ICASSP 2017*, 2017, pp. 5220–5224.

[39] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint:1510.08484*, 2015.

[40] S. Saleem, A. Dilawari, and U. G. Khan, "Spoofed voice detection using dense features of stft and mdct spectrograms," in *2021 International Conference on Artificial Intelligence (ICAI)*, 2021, pp. 56–61.

[41] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. INTERSPEECH 2019*, 2019, pp. 2613–2617.

[42] H. Tak, M. Kamble, J. Patino, M. Todisco, and N. Evans, "RawBoost: A raw data boosting and augmentation method applied to automatic speaker verification anti-spoofing," in *Proc. ICASSP 2022*, 2022, pp. 6382–6386.

[43] X. Wang and J. Yamagishi, "Investigating active-learning-based training data selection for speech spoofing countermeasure," in *2022 IEEE Spoken Language Technology Workshop (SLT)*, 2023, pp. 585–592.

[44] ——, "Spoofed training data for speech spoofing countermeasure can be efficiently created using neural vocoders," in *Proc. ICASSP 2023*, 2023, pp. 1–5.

[45] N. Müller, P. Czempin, F. Diekmann, A. Froghyar, and K. Böttinger, "Does audio deepfake detection generalize?" in *Proc. INTERSPEECH 2022*, 2022, pp. 2783–2787.

[46] J.-w. Jung, H. Tak, H.-j. Shim, H.-S. Heo, B.-J. Lee, S.-W. Chung, H.-G. Kang, H.-J. Yu, N. Evans, and T. Kinnunen, "SASV Challenge 2022: A spoofing aware speaker verification challenge evaluation plan," *arXiv preprint:2201.10283*, 2022.

[47] A. Alenin, N. Torgashov, A. Okhotnikov, R. Makarov, and I. Yakovlev, "A subnetwork approach for spoofing aware speaker verification," in *Proc. INTERSPEECH 2022*, 2022, pp. 2888–2892.

[48] X. Wang, X. Qin, Y. Wang, Y. Xu, and M. Li, "The DKU-OPPO system for the 2022 spoofing-aware speaker verification challenge," in *Proc. INTERSPEECH 2022*, 2022, pp. 4396–4400.

[49] J.-H. Choi, J.-Y. Yang, Y. Jeoung, and J.-H. Chang, "HYU submission for the SASV challenge 2022: Reforming speaker embeddings with spoofing-aware conditioning," in *Proc. INTERSPEECH 2022*, 2022, pp. 2873–2877.

[50] L. Zhang, Y. Li, H. Zhao, and L. Xie, "Backend ensemble for speaker verification and spoofing countermeasure," in *Proc. INTERSPEECH 2022*, 2022, pp. 4381–4385.

[51] J. Heo, J.-h. Kim, and H.-s. Shin, "Two methods for spoofing-aware speaker verification: Multi-layer perceptron score fusion model and integrated embedding projector," in *Proc. INTERSPEECH 2022*, 2022, pp. 2878–2882.

[52] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[53] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. ICASSP 2014*, 2014, pp. 1695–1699.

[54] H. Zeinali, L. Burget, H. Sameti, O. Glembek, and O. Plchot, "Deep neural networks and hidden markov models in i-vector-based text-dependent speaker verification," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2016)*, vol. 2016, 2016, pp. 24–30.

[55] M. McLaren, L. Ferrer, and A. Lawson, "Exploring the role of phonetic bottleneck features for speaker and language recognition," in *Proc. ICASSP 2016*, 2016, pp. 5575–5579.

[56] A. Luo, E. Li, Y. Liu, X. Kang, and Z. J. Wang, "A capsule network based approach for detection of audio spoofing attacks," in *Proc. ICASSP 2021*, 2021, pp. 6359–6363.

[57] Y. Zhang, F. Jiang, and Z. Duan, "One-class learning towards synthetic voice spoofing detection," *IEEE Signal Processing Letters*, vol. 28, pp. 937–941, 2021.

[58] J.-w. Jung, H.-s. Heo, J.-h. Kim, H.-j. Shim, and H.-j. Yu, "Rawnet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," in *Proc. INTERSPEECH 2019*, 2019, pp. 1268–1272.

[59] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," *IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028, 2018.

[60] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher, "End-to-end anti-spoofing with RawNet2," in *Proc. ICASSP 2020*, 2020, pp. 6369–6373.

[61] H. Tak, J.-w. Jung, J. Patino, M. Todisco, M. Kamble, Massimiliano, and N. Evans, "End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection," in *Proc. ASVspoof 2021 Workshop*, 2021, pp. 1–8.

[62] H. Tak, J. Patino, A. Nautsch, N. Evans, and M. Todisco, "An explainability study of the Constant Q Cepstral Coefficient spoofing countermeasure for automatic speaker verification," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2020)*, 2020, pp. 333–340.

[63] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[64] B. Halpern, F. Kelly, B. van Son, and A. Alexander, "Residual networks for resisting noise: Analysis of an embeddings-based spoofing countermeasure," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2020)*, 2020, pp. 326–332.

[65] B. Chettri, S. Mishra, B. L. Sturm, and E. Benetos, "Analysing the predictions of a CNN-based replay spoofing detection system," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 92–97.

[66] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?" Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[67] L. S. Shapley, "A value of n-person games," *Contributions to the Theory of Games*, pp. 307–317, 1953.

[68] S. M. Lundberg, S.-i. Lee, and D. Fohr, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[69] S. Sivasankaran, E. Vincent, and D. Fohr, "Explaining deep learning models for speech enhancement," in *Proc. INTERSPEECH 2021*, 2021, pp. 696–700.

[70] X. Zhou, D. Garcia-Romero, R. Duraiswami, C. Espy-Wilson, and S. Shamma, "Linear versus mel frequency cepstral coefficients for speaker recognition," in *2011 IEEE workshop on automatic speech recognition & understanding*, 2011, pp. 559–564.

[71] J. Martinez, H. Perez, E. Escamilla, and M. M. Suzuki, "Speaker recognition using Mel frequency Cepstral Coefficients (MFCC) and Vector quantization (VQ) techniques," in *Conielecomp 2012, 22nd International conference on electrical communications and computers*, 2012, pp. 248–251.

[72] J.-w. Jung, S.-b. Kim, H.-j. Shim, J.-h. Kim, and H.-j. Yu, "Improved RawNet with feature map scaling for text-independent speaker verification using raw waveforms," in *Proc. INTERSPEECH 2020*, 2020, pp. 1496–1500.

[73] P. Safari, M. India, and J. Hernando, "Self-attention encoding and pooling for speaker recognition," *Proc. INTERSPEECH 2020*, pp. 941–945, 2020.

[74] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," *Proc. INTERSPEECH 2019*, pp. 4305–4309, 2019.

[75] Y. Li, F. Gao, Z. Ou, and J. Sun, "Angular softmax loss for end-to-end speaker verification," in *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2018, pp. 190–194.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[77] T. Chen, A. Kumar, P. Nagarsheth, G. Sivaraman, and E. Khoury, "Generalization of audio deepfake detection," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2020)*, 2020, pp. 132–137.

[78] J.-w. Jung, H.-S. Heo, H. Tak, H.-j. Shim, J. S. Chung, B.-J. Lee, H.-J. Yu, and N. Evans, "AASIST: Audio anti-spoofing using integrated spectro-temporal graph attention networks," in *Proc. ICASSP 2022*, 2022, pp. 6367–6371.

[79] H. Tak, M. Todisco, X. Wang, J.-w. Jung, J. Yamagishi, and N. Evans, "Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)*, 2022, pp. 112–119.

[80] X. Wang and J. Yamagishi, "Investigating self-supervised front ends for speech spoofing countermeasures," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)*, 2022, pp. 100–106.

[81] Y. Ma, Z. Ren, and S. Xu, "RW-Resnet: A novel speech anti-spoofing model using raw waveform," in *Proc. INTERSPEECH 2021*, 2021, pp. 4144–4148.

[82] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[83] G. Valenti, H. Delgado, M. Todisco, N. W. Evans, and L. Pilati, "An end-to-end spoofing countermeasure for automatic speaker verification using evolving recurrent neural networks," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2018)*, 2018, pp. 288–295.

[84] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *International Conference on Learning Representations*, 2017.

[85] T. Elsken, J. H. Metzen, F. Hutter *et al.*, "Neural architecture search: A survey." *Machine Learnign Research*, vol. 20, no. 55, pp. 1–21, 2019.

[86] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[87] T. Mo, Y. Yu, M. Salameh, D. Niu, and S. Jui, "Neural architecture search for keyword spotting," in *Proc. INTERSPEECH 2020*, 2020, pp. 1982–1986.

[88] Yi-Chen Chen and Jui-Yang Hsu and Cheng-Kuang Lee and Hung-yi Lee, "DARTS-ASR: Differentiable architecture search for multilingual speech recognition and adaptation," in *Proc. INTERSPEECH 2020*, 2020, pp. 1803–1807.

[89] S. Ding, T. Chen, X. Gong, W. Zha, and Z. Wang, "AutoSpeech: Neural architecture search for speaker recognition," in *Proc. INTERSPEECH 2020*, 2020, pp. 916–920.

[90] T. Kinnunen, H. Delgado, N. Evans, K. A. Lee, V. Vestman, A. Nautsch, M. Todisco, X. Wang, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "Tandem assessment of spoofing countermeasures and automatic speaker verification: Fundamentals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2195–2210, 2020.

[91] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, N. Evans, M. Sahidullah, V. Vestman, T. Kinnunen, K. A. Lee *et al.*, "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," *Computer Speech & Language*, vol. 64, p. 101114, 2020.

[92] T. Kinnunen, K. A. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi, and D. Reynolds, "t-DCF: A detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2018)*, 2018, pp. 312–319.

[93] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, Y. Bengio and Y. LeCun, Eds., 2015.

[94] Z. Yu, C. Zhao, Z. Wang, Y. Qin, Z. Su, X. Li, F. Zhou, and G. Zhao, "Searching central difference convolutional networks for face anti-spoofing," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5294–5304.

[95] X. Li, N. Li, C. Weng, X. Liu, D. Su, D. Yu, and H. Meng, "Replay and synthetic speech detection with Res2net architecture," in *Proc. ICASSP 2021*, 2021, pp. 6354–6358.

[96] S. Liu, H. Wu, H.-y. Lee, and H. Meng, "Adversarial attacks on spoofing countermeasures of automatic speaker verification," in *2019 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2019, pp. 312–319.

[97] M. Alzantot, Z. Wang, and M. B. Srivastava, "Deep residual neural networks for audio spoofing detection," in *Proc. INTERSPEECH 2019*, 2019, pp. 1078–1082.

[98] H. Dinkel, N. Chen, Y. Qian, and K. Yu, "End-to-end spoofing detection with raw waveform CLDNNS," in *Proc. ICASSP 2017*, 2017, pp. 4860–4864.

[99] W. Ge, M. Panariello, J. Patino, M. Todisco, and N. Evans, "Partially-connected differentiable architecture search for deepfake and spoofing detection," in *Proc. INTERSPEECH 2021*, 2021, pp. 4319–4323.

[100] S. Cai, Y. Shu, G. Chen, B. C. Ooi, W. Wang, and M. Zhang, "Effective and efficient dropout for deep convolutional neural networks," *arXiv preprint arXiv:1904.03392*, 2019.

[101] S. Hou and Z. Wang, "Weighted channel dropout for regularization of deep convolutional neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8425–8432.

[102] H. Wang, Y. Zou, and W. Wang, "SpecAugment++: A hidden space data augmentation method for acoustic scene classification," *Proc. INTERSPEECH 2021*, pp. 551–555, 2021.

[103] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[104] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[105] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1294–1303.

[106] A. Yang, P. M. Esperança, and F. M. Carlucci, "NAS evaluation is frustratingly hard," in *International Conference on Learning Representations*, 2020.

[107] X. Zhang, R. Zhao, J. Yan, M. Gao, Y. Qiao, X. Wang, and H. Li, "P2sgrad: Refined gradients for optimizing deep face models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9906–9914.

[108] K. Tan, J. Chen, and D. Wang, "Gated residual networks with dilated convolutions for supervised speech separation," in *Proc. ICASSP 2018*, 2018, pp. 21–25.

[109] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR*, 2016.

[110] H. Tak, J.-w. Jung, J. Patino, M. Kamble, M. Todisco, and N. Evans, "End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection," in *Proc. ASVspoof 2021 Workshop*, 2021, pp. 1–8.

[111] A. Nautsch, X. Wang, N. Evans, T. Kinnunen, V. Vestman, M. Todisco, H. Delgado, M. Sahidullah, J. Yamagishi, and K. A. Lee, "ASVspoof 2019: Spoofing countermeasures for the detection of synthesized, converted and replayed speech," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, no. 2, pp. 252–265, 2021.

[112] M. Christoph, "Interpretable machine learning: A guide for making black box models explainable," 2020.

[113] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 2017, pp. 3145–3153.

[114] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.

[115] S. Becker, M. Ackermann, S. Lapuschkin, K. Müller, and W. Samek, "Interpreting and explaining deep neural networks for classification of audio signals," *CoRR*, vol. abs/1807.03418, 2018.

[116] C. Sun, S. Jia, S. Hou, and S. Lyu, "AI-synthesized voice detection using neural vocoder artifacts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 904–912.

[117] R. Ranjan, M. Vatsa, and R. Singh, "Statnet: Spectral and temporal features based multi-task network for audio spoofing detection," in *2022 IEEE International Joint Conference on Biometrics (IJCB)*, 2022, pp. 1–9.

[118] N. Subramani and D. Rao, "Learning efficient representations for fake speech detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5859–5866.

[119] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *arXiv preprint arXiv:1609.03499*, 2016.

[120] E. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[121] M. Schröder, M. Charfuelan, S. Pammi, and I. Steiner, "Open source voice creation toolkit for the MARY TTS Platform," in *Proc. INTERPSEECH 2011*, 2011, pp. 3253–3256.

[122] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "ASVspoof 2015: The first automatic speaker verification spoofing and countermeasures challenge," in *Proc. INTERPSEECH 2015*, 2015, pp. 2037–2041.

[123] D. Matrouf, J. Bonastre, and C. Fredouille, "Effect of speech transformation on impostor acceptance," in *Proc. ICASSP 2006*, 2006, pp. 933–936.

[124] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *International Conference on Machine Learning*, 2021, pp. 5530–5540.

[125] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. International Conference on Learning Representations*, 2014.

[126] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. ICML*, 2015, pp. 1530–1538.

[127] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," in *Proc. NIPS*, 2019, pp. 7511–7522.

[128] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. International Conference on Learning Representations*, 2018.

[129] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goya *et al.*, "XLS-R: Self-supervised cross-lingual speech representation learning at scale," in *Proc. INTERSPEECH 2022*, 2022, pp. 2278–2282.

[130] A. Mohamed, H.-y. Lee, L. Borgholt, J. D. Havtorn, J. Edin, C. Igel *et al.*, "Self-supervised speech representation learning: A review," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1179–1210, 2022.

[131] J. Yamagishi, C. Veaux, and K. MacDonald, "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92)," *University of Edinburgh. The Centre for Speech Technology Research*, 2019.

[132] Z. Bai and X.-L. Zhang, "Speaker recognition based on deep learning: An overview," *Neural Networks*, vol. 140, pp. 65–99, 2021.

[133] Z. Chen, S. Chen, Y. Wu, Y. Qian, C. Wang, S. Liu, Y. Qian, and M. Zeng, "Large-scale self-supervised speech representation learning for automatic speaker verification," in *Proc. ICASSP 2022*, 2022, pp. 6147–6151.

[134] J.-w. Jung, Y. J. Kim, H.-S. Heo, B.-J. Lee, Y. Kwon, and J. S. Chung, "Pushing the limits of raw waveform speaker recognition," in *Proc. INTERSPEECH 2022*, 2022, pp. 2228–2232.

[135] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 022–17 033, 2020.

[136] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, "Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone," in *International Conference on Machine Learning*, 2022, pp. 2709–2720.

[137] B. Sisman, J. Yamagishi, S. King, and H. Li, "An overview of voice conversion and its challenges: From statistical modeling to deep learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 132–157, 2020.

[138] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (SITW) speaker recognition database," in *Proc. INTERSPEECH 2016*, 2016, pp. 818–822.

[139] J. Yi, R. Fu, J. Tao, S. Nie, H. Ma, C. Wang, T. Wang, Z. Tian, Y. Bai, C. Fan, S. Liang, S. Wang, S. Zhang, X. Yan, L. Xu, Z. Wen, and H. Li, "ADD 2022: The first audio deep synthesis detection challenge," in *Proc. ICASSP 2022*, 2022, pp. 9216–9220.

[140] E. Khoury, T. Kinnunen, A. Sizov, Z. Wu, and S. Marcel, "Introducing i-vectors for joint anti-spoofing and speaker verification," in *Proc. INTERSPEECH 2014*, 2014, pp. 61–65.

[141] J.-w. Jung, H. Tak, H.-j. Shim, H.-S. Heo, B.-J. Lee, S.-W. Chung, H.-J. Yu, N. Evans, and T. Kinnunen, "SASV 2022: The first spoofing-aware speaker verification challenge," in *Proc. INTERSPEECH 2022*, 2022, pp. 2893–2897.

[142] Y. Kwon, H.-S. Heo, B.-J. Lee, and J. S. Chung, "The ins and outs of speaker recognition: Lessons from VoxSRC 2020," in *Proc. ICASSP 2021*, 2021, pp. 5809–5813.

[143] F. Alegre, A. Amehraye, and N. Evans, "A one-class classification approach to generalised speaker verification spoofing countermeasures using local binary patterns," in *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 2013, pp. 1–8.

[144] W. H. Kang, J. Alam, and A. Fathan, "End-to-end framework for spoof-aware speaker verification," in *Proc. INTERSPEECH 2022*, 2022, pp. 4362–4366.

[145] B. T. Ta, T. L. Nguyen, D. S. Dang *et al.*, "A multi-task conformer for spoofing aware speaker verification," in *Proceedings of the IEEE Ninth International Conference on Communications and Electronics*, 2022, pp. 306–310.

[146] Z. Teng, Q. Fu, J. White, M. E. Powell, and D. C. Schmidt, "SA-SASV: An end-to-end spoof-aggregated spoofing-aware speaker verification system," in *Proc. INTERSPEECH 2022*, 2022, pp. 4391–4395.

[147] Y. Zhang, Z. Li, W. Wang, and P. Zhang, "SASV based on pre-trained ASV system and integrated scoring module," in *Proc. INTERSPEECH 2022*, 2022, pp. 4376–4380.

[148] P. Zhang, P. Hu, and X. Zhang, "Norm-constrained score-level ensemble for spoofing aware speaker verification," in *Proc. INTERSPEECH 2022*, 2022, pp. 4371–4375.

[149] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[150] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Proc. INTERSPEECH 2018*, 2018, pp. 2252–2256.

[151] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2593–2601.

[152] H.-j. Shim, H. Tak, X. Liu, H.-S. Heo, J.-w. Jung, J. S. Chung, S.-W. Chung *et al.*, "Baseline systems for the first spoofing-aware speaker verification challenge: score and embedding fusion," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)*, 2022, pp. 330–337.

[153] S. Ding, T. Chen, X. Gong, W. Zha, and Z. Wang, "AutoSpeech: Neural architecture search for speaker recognition," in *Proc. INTERSPEECH 2020*, 2020, pp. 916–920.