Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Few-Shot Learning and Learnable Frontends for Remote Monitoring of Bird Populations

Mark William Anderson

A dissertation submitted in fulfilment

of the requirements for the degree of

Doctor of Philosophy

Trinity College Dublin, the University of Dublin

2024

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Mark William Anderson

May 2, 2024

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Mark William Anderson

May 2, 2024

Acknowledgements

The submission of this thesis marks the end of an almost decade-long journey here at Trinity College, and specifically the Department of Electronic & Electrical Engineering. I have many *many* people to thank!

First, I would like to thank my supervisor Dr. Naomi Harte, for providing continuous guidance and support throughout my PhD. Naomi your patience, knowledge and good humour have enabled me to develop as a researcher, and to stay sane while doing so! I am incredibly grateful for the opportunity to collaborate with you over these past few years.

I want to thank Dr. Tomi Kinnunen for hosting me during a productive and enlightening visit to the University of Eastern Finland. Our long, very nerdy chats at lunch, covering topics such as signal processing, AI, and music, are cherished memories.

I would also like to thank the academic, technical and administrative staff of the department, for both encouragement during my research, and outside of it. Special thanks to Dr. Sébastien Le Maguer, my partner in productive procrastination, to Dr. Anil Kokaram for engaging discussions on signal processing, cinema, and cricket, and to Dr. Harun Šiljak, for not discussing research at all. Thank you to Shane Hunt, Mark Linnane, Cormac Molloy, Eugene O'Rourke, John Squires and Michael O'Riordan, not only for all your help throughout the years, but also for plenty of good humour and chat.

Thank you and good luck to all my fellow PhD researchers in Sigmedia (and our neighbours in Dr. Arman Farhang's lab). I want to wish the best of luck to Sam Kotey, Ed Storey, Sam O'Connor Russell, Clément Bled, Darren Ramsook, Vibhoothi, Meegan Gower, Xin Shu, Conall Daly, Uditangshu Aurangabadkar, Julien Zouein and Zhaofeng Lin. A special thank

you and good luck to my dear friend Ayushi Pandey, who embarked on this journey alongside me. Ayushi, no matter what happens, I think you can truly call yourself a Dubliner at this point.

I would also like to express my gratitude to (now Dr.) DJ Ringis. Your ability to easily distract me and your infectious enthusiasm for sports, games, and politics has been a joy the last half decade. Thank you!

This wouldn't have been possible without the love and support of my parents, Barbara & David. Their encouragement and patience with a young "engineer", occupying space and breaking more things than he got to work, helped lead me to pursuing engineering and ultimately, finishing this work. Their willingness to hear me complain (when needed) for four years has helped immensely.

Finally I want to thank Ellie. You have supported and helped me more than you know, and in more ways than I could ever put into words (but I'm going to keep trying). Thank you for everything.

Abstract

In response to changing ecological and environmental factors, automated monitoring of bird populations has become imperative for conservation efforts and as an indicator of change in its own right. This is particularly crucial in remote areas where manual observation is challenging. Bird vocalisations are highly suitable for population monitoring, particularly in environments where visual analysis is impractical. Automating the detection of bird activity would allow ornithologists and conservationists more time for in-depth analysis of the data.

Essential to this effort is the development of high-performing, efficient systems capable of operating on low-resource devices which are suitable for field deployment. However, the limited availability of extensive, temporally detailed, fully annotated datasets for bird audio poses a challenge for generalisation. Few-shot learning, especially in the context of sound event detection, emerges as a promising solution to address the scarcity of data. It offers a potentially lightweight approach to monitoring bird populations through their vocalisations.

Additionally, in the age of deep learning, the predominant input features to systems are time-frequency representations of the audio. Recent advancements in learnable frontends have enabled systems to learn from raw waveforms, learning new filterbanks for time-frequency representations or learnable compression which is applied to existing representations. The incorporation of learnable frontends into deep learning systems enables the learning of improved audio representations directly from the audio data.

This thesis seeks to advance automatic and remote bird population monitoring through the development of activity detection models and the integration of learnable frontends into bird audio analysis. Features and classifiers are explored to determine their suitability for bird

audio monitoring, considering both performance and computational efficiency. This includes development of the AMPS feature set, derived from amplitude modulation, pitch and spectral features, which is suitable for use with low-resource classifiers. Additionally, a few-shot learning system for bioacoustic activity detection is developed as a system that can generalise from few labelled examples and is potentially suitable for field deployment. We incorporate learnable frontends into this system, yielding a relative 25% increase in F1-score over static time-frequency representations.

Furthermore, this thesis benchmarks traditional fixed-parameter frontends against a new generation of learnable frontends when applied to bird audio. We observe that Per-Channel Energy Normalisation is the best overall performer and that in general learnable frontends significantly outperform traditional methods. While the integration of learnable frontends enhances overall performance, those employing learnable filterbanks exhibit sensitivity to initialisation. We characterise the sensitivity of a learnable filterbank to its initialisation using several strategies on two audio tasks: voice activity detection and bird species identification. The limited movement of the filters from their initialisation suggests that alternative optimisation strategies may allow a learnable filterbank to reach better overall performance. To address this, we propose two mitigation strategies which modify the training strategy to encourage filter movement. While yielding inconclusive results, these attempts serve as a preliminary step for future research on the filterbank initialisation problem.

Contents

List of Tables

List of Figures

List of Acronyms

AGC Automatic Gain Control

ANOVA Analysis of Variance

ARU Automatic Recording Units

AUC Area Under Receiver Operating Curve

BAD Bird Activity Detection

BSID Bird Species Identification

CNN Convolutional Neural Networks

CQT Constant-Q Transform

CRNN Convolutional Recurrent Neural Networks

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DNN Deep Neural Network

DRC Dynamic Range Compression

eLEAF Efficient Learnable Audio Frontend

FFT Fast Fourier Transform

FSL Few-Shot Learning

FWHM Full-Width at Half Maximum

GAN Generative Adversarial Networks

GMM Gaussian Mixture Models

HMM Hidden Markov Models

JSD Jensen-Shannon distance

LEAF Learnable Audio Frontend

MFCC Mel-Frequency Cepstral Coefficients

PCA Principal Component Analysis

PCEN Per-Channel Energy Normalisation

SGD Stochastic Gradient Descent

STA-VAD Spectro-Temporal Voice Activity Detection

STFT Short-Time Fourier Transform

STRF Spectro-Temporal Filters

SVM Support Vector Machines

t-SNE t-Distributed Stochastic Neighbour Embedding

TD Time-Domain Filter Banks

TF Time-Frequency

TIM Transductive Information Maximization

VAD Voice Activity Detection

Chapter 1

Introduction

1.1 Monitoring Bird Populations through Audio

1.1.1 The Importance of Monitoring Bird Populations

Human impact on the Earth's ecosystems has led to the eradication and modification of natural habitats [204]. This impact has accelerated the extinction of numerous species, resulting in a reduction in biodiversity [29]. Long-term monitoring of bird populations is of increasing importance both for scientific research and conservation as ecological and environmental factors change [83, 196]. Moreover, it serves as an indicator of change in its own right [55]. Automatic identification of bird activity and species is crucial for effective conservation.

Traditionally, monitoring bird populations involves labour-intensive fieldwork conducted by teams of ornithologists and related experts. This approach is not only expensive and time-consuming but also potentially disruptive to natural habitats. Although visual inspection is used, identification of activity and species has utilised bird audio since the 1950's [194]. Audio analysis is particularly suitable for monitoring bird populations, as many species are distinguishable by their vocalisations [26]. Therefore, even when visual identification is impractical, audio analysis remains a sufficient tool for monitoring populations.

The widespread application of soundscape analysis to bird audio can enable automatic

analysis and assist ornithologists, ecologists, and conservationists in their endeavours. Research in bioacoustics, including bird vocalisation analysis, has experienced significant growth in the last decade, primarily driven by the advent of deep learning [184]. The development of high-performing, efficient systems capable of running on constrained devices suitable for field deployment is crucial for enhancing conservation efforts, considering both raw material and computational power constraints [113].

1.1.2 Difficulties with Monitoring Bird Populations through their Vocalisations

Monitoring of bird populations through their vocalisations presents challenges. While large deep learning models are effective, there is a growing need to deploy systems on edge devices with hardware constraints, which is crucial for the widespread adoption of computational bioacoustic monitoring [113]. In certain deployment scenarios, a communication link may be impractical, preventing the uploading of extensive data to the cloud for analysis. Additionally, some processing may be preferable on-device, even with a viable connection or sufficient storage space, serving as an initial step in filtering relevant data and providing preliminary analysis to researchers. Designing systems capable of running on constrained hardware necessitates compromises, such as opting for smaller models, extensive optimisation, or embracing new approaches like Few-Shot Learning (FSL).

A persistent challenge in bioacoustic projects, including those focused on bird audio, is the scarcity of large labelled datasets [184]. Many available datasets stem from citizen science initiatives or public repositories, introducing variations in equipment, audio quality, recording conditions, and types of recordings. These range from purposeful, directed recordings of one bird to omni-directional soundscape recordings, in addition to variations in near-field or far-field subjects. Moreover, numerous datasets lack adequate temporal resolution, often having low-resolution annotations or clip-level annotations with no temporal details. Unbalanced datasets, with an abundance of examples for certain classes and very few for others, are common. This imbalance is particularly noticeable in datasets containing rare or endangered species, which are the species most in need of monitoring.

Environmental noise poses a significant challenge in the analysis of bird audio, particularly non-stationary noise sources that complicate consistent analysis. Eliminating noise without compromising bird audio proves difficult, especially when faced with in-band noise like insects, wind, and non-target species. While some datasets are curated to maintain a certain signal-to-noise ratio (SNR), many sourced from public repositories are not curated in this way. Moreover, hand-picked low-noise examples may not accurately reflect real-world situations, where birds coexist with other animals and variable weather conditions exist. Although low-noise datasets are beneficial in training, the inclusion of low SNR audio in the model is crucial for eventual deployment.

Another challenge lies in feature extraction and representation. Bird audio analysis, influenced by human speech analysis, often adopts mel scale-derived features like Mel-Frequency Cepstral Coefficients (MFCC) or more recently, log-mel spectrograms. However, the reliance on the mel scale, rooted in human perception, may not be optimal for bird audio tasks, as highlighted in studies on bird audio monitoring literature [7, 216]. While there is an understanding of how birds perceive songs [25], there is currently no equivalent of the mel scale for songbirds, and such a scale would likely be species-specific.

Monitoring birds through audio brings about challenges: the demand for systems compatible with modest or low-resource hardware; issues related to dataset annotation and recording consistency; and environmental noise contamination. Section 1.2 below states the goals and research questions of this thesis in the context of the importance of monitoring birds through their audio, and the problems discussed above.

1.2 Thesis Statement

This thesis seeks to advance automatic and remote monitoring of bird populations using their vocalisations. This is accomplished by exploring and developing low-resource models and integrating learnable frontends into the analysis of bird audio. Rather than serving as an engineering constraint, the development of low-resource classifiers is a guiding principle that facilitates future work on the deployment of such systems. To achieve these objectives, the thesis poses the following research questions:

RQ1 What machine learning approaches are appropriate for automatic and remote monitoring of bird populations on low-resource hardware?

- **RQ1.1** What features and classifiers are suitable for bird audio monitoring, considering both accuracy and computational efficiency?
- **RQ1.2** How can few-shot learning approaches be effectively applied to bird audio monitoring?

RQ2 What is the effectiveness of learnable frontends in audio deep learning systems, particularly in the context of bird audio?

- **RQ2.1** How do different learnable frontends impact the performance of deep learning with bird audio?
- **RQ2.2** To what extent are learnable frontends sensitive to their intialisation?

The first part of this thesis, Chapters 3 and 4, primarily addresses **RQ1** and its two sub-questions. These chapters explore low-resource classifiers using a feature set grounded in an understanding of bird vocalisation production, and the usage of few-shot learning approaches. Findings in Chapter 4 suggest that improving a few-shot activity detector would be best achieved by focusing on improved feature representation. This leads into the second part of the thesis, Chapters 5 and 6, which address **RQ2** and its sub-questions, as well as returning to **RQ1.2**. Chapter 5 evaluates the effectiveness of learnable frontends on a bird audio task, as well as identifying and characterising the sensitivity of learnable filterbanks to their initialisation. Chapter 6 proposes some mitigation strategies for this problem, and returns to few-shot learning with the addition of learnable frontends. More detailed outlines of each chapter can be found below in Section 1.3.

1.3 Thesis Outline

This thesis is outlined as follows.

Chapter 2: Literature Review

This chapter offers an introduction to many key topics. Section 2.1 covers birdsong production and perception, including its signal structure and properties. Section 2.2 provides insights into recording methods used and introduces some tasks in bird audio monitoring. Chapters 4 and 6 utilise Few-Shot Learning, so Section 2.3 provides an overview of FSL concepts, approaches, and applications in audio and bioacoustics. In Chapters 5 and 6, learnable frontends are investigated, and Section 2.4 covers frequency-domain-based and time-domain-based filterbanks, common problems with learnable filterbanks, and learnable compression. Finally, Section 2.5 covers the datasets used in this thesis, and Section 2.6 discusses how the literature connects to this work.

Chapter 3: Low Resource Bird Activity Detection with AMPS

This chapter explores low-resource classifiers and features for bird activity detection on embedded devices used in long-term bird population monitoring. These features include low-level spectral parameters, statistical moments on pitch samples, and features based on amplitude modulation (see Section 3.1.3). In particular, this chapter introduces amplitude modulation-based features inspired by work from the Institute of Acoustics, drawing upon the correlation between a bird species' vocalisation frequency range and their trill rate. Performance is assessed on several lightweight classifiers using the NIPS4Bplus dataset, compared against both a CNN-based detector, and MFCC features (Section 3.3). The experiments show that random forest classifiers perform best on this task, achieving an accuracy of 0.721 and an F1-Score of 0.604. The findings demonstrate that equal or better performance can be attained with lower computational demands.

Chapter 4: Bioacoustic Event Detection with Prototypical Networks

This chapter covers participation in the DCASE2021 Few-Shot Bioacoustic Sound Event Detection Challenge. Section 4.2.1 outlines the challenge itself. Prototypical Networks are employed as our chosen FSL method, with a more detailed explanation of their operation in Section 4.1. The chapter discusses the network architecture, feature extraction methods, and data augmentation applied to the challenge dataset. It also includes comparisons with the

challenge's baseline networks (Section 4.2.4) and the other challenge entries (Section 4.2.5). Furthermore, the chapter conducts additional analysis of the resulting embedding space and proposes and evaluates methods for enhancing system performance in Section 4.3. This analysis led to the conclusion that more significant improvements are achievable by enhancing data representations, rather than making marginal adjustments to network architecture and training.

Chapter 5: Learnable Frontends and the Filterbank Initialisation Problem

The choice of acoustic frontend directly impacts system performance. In this chapter, we compare traditional fixed-parameter frontends with new learnable frontends for bird audio detection using data from the DCASE2018 BAD Challenge (Section 5.3). Section 5.2 provides detailed explanations of the frontends. Results in Section 5.3.4 show that Per-Channel Energy Normalisation is the best overall performs the best with 89.9%, and generally, learnable frontends outperform traditional methods. However, previous studies have noted that frontends using learnable filterbanks do not differ substantially from their initialisation. In Section 5.4 we investigate filterbank initialisation and the sensitivity of a learnable filterbank to its initialisation on two audio tasks: voice activity detection and bird species identification. Analysing both Jensen-Shannon distance and filterbank shape before and after training (Section 5.4.3), we show that although performance is overall improved, the filterbanks exhibit strong sensitivity to their initialisation.

Chapter 6: Mitigation Strategies for the Filterbank Initialisation Problem and Returning to Few-Shot Learning

This chapter proposes mitigation strategies to combat the filterbank initialisation problem. Results from Section 5.4.3 are compared to results using three mitigation strategies: an alternating training method, where the frontend and classifier are trained alternately; separate optimisation, where a higher learning rate is used to train the frontend; and a strategy combining both of these methods. The results indicate that a higher learning rate of the frontend compared to the classifier can increase performance, but that the sensitivity to initialisation remains. We also apply learnable frontends to the Prototypical Network in

Chapter 4, leading to a significant increase in performance. This adaptation results in a 25% relative improvement in F1-Score compared to the original system.

Chapter 7: Conclusion

This chapter draws together conclusions from the previous chapters, and discusses their significance and impact. Challenges which remain and have been introduced as a result of the work done are discussed and possible directions for future work are outlined.

1.4 Contributions

As outlined in Section 1.2, this thesis aims to advance automatic and remote monitoring of bird populations using their vocalisations. Section 1.2 introduces two primary research questions, each accompanied by two sub-questions, providing the foundation for the original contributions to knowledge within this thesis. After outlining the thesis structure in Section 1.3, which provides context for the organisation of the thesis, these contributions to knowledge can be summarised as follows:

1. **The AMPS Feature Set**

   AMPS, tailored for low-resource classifiers like random forests, leverages the relationship between amplitude modulation and frequency modulation in birdsong, alongside incorporating low-level spectral features. AMPS surpasses MFCCs on a species agnostic bird activity detection task, and addresses **RQ1.1**. This contribution arises from Chapter 3, which presents the feature extraction pipeline and comparative evaluation against alternative methods, and was presented [4] in SiPS 2021.

2. **Dedicated Evaluation of Learnable Frontends for Bird Audio**

   With the rise of audio deep learning systems using 'off-the-shelf' learnable frontends, evaluating existing approaches and their application to bird audio was essential. Addressing **RQ2.1**, the work in Section 5.3, and the associated publication [3] presented in IWAENC 2022, is the most comprehensive investigation to date on the suitability of learnable frontends in bird audio. The evaluation indicates a general

performance boost with learnable frontends. Per-Channel Energy Normalisation outperforms all other methods and is the recommended learnable frontend for bird audio currently, especially over static compression methods such as logarithmic compression.

3. **Characterisation of the Filterbank Initialisation Problem**

   Existing literature on learnable frontends with learnable filterbanks notes a lack of learning in the filterbank component. The same literature tends to be dismissive of this issue, attributing it to the belief that the initialisation was already well-suited for the task. Section 5.4 presents a systematic study which characterises and quantifies the filterbank initialisation problem, originally presented [5] in ICASSP 2023. The results in Section 5.4 provide novel insights into the shortcomings of learnable filterbank-based frontends, revealing their sensitivity to initialisation, addressing **RQ2.2**.

4. **Mitigation Strategies for the Filterbank Initialisation Problem**

   Two modifications to the training strategy are proposed to combat the effects of the filterbank initialisation problem. These strategies involve alternating training between the frontend and backend parameters, as well as employing separate optimisers for each parameter group. Section 6.1 provides a comprehensive evaluation of these strategies using the same tasks employed to characterise the initialisation problem. Using separate optimisers with a higher learning rate for the frontend enhances performance and filter movement in human speech tasks, offering greater flexibility in training. However, the effectiveness of these strategies on other tasks and models is inconclusive, and they serve as a starting point for future work on the filterbank initialisation problem.

5. **Few-Shot Learning with Learnable Frontends for Bioacoustic Activity Detection**

   We developed a prototypical network-based [177] few-shot learning system for bioacoustic activity detection, the first to successfully incorporate fully-trainable frontends within the model. Chapter 4 details the initial development and evaluation of the system, initially submitted [2] as an entry to the DCASE2021 Few-Shot Bioacoustic Sound Event Detection Challenge. Section 6.2 delves into the

incorporation of learnable frontends into the model, and discusses the performance increases from doing so. This system addresses **RQ1.2**, as a low-resource, few-shot learning-based approach to bird audio monitoring.

1.5 Publications

As mentioned above, portions of the work described in this thesis have appeared in the following publications [2, 3, 4, 5]. These publications are listed here for convenience and are discussed in relevant sections.

- M. Anderson, J. Kennedy, and N. Harte, "Low Resource Species Agnostic Bird Activity Detection," in 2021 IEEE Workshop on Signal Processing Systems (SiPS), 2021, pp. 34–39.
- M. Anderson and N. Harte, "Bioacoustic Event Detection with Prototypical Networks and Data Augmentation," DCASE2021 Challenge, Tech. Rep., Jun. 2021
- M. Anderson and N. Harte, "Learnable Acoustic Frontends in Bird Activity Detection," in 2022 International Workshop on Acoustic Signal Enhancement (IWAENC), 2022, pp. 1–5.
- M. Anderson, T. Kinnunen, and N. Harte, "Learnable Frontends that do not Learn: Quantifying Sensitivity to Filterbank Initialisation," in ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–5.

Chapter 2

Literature Review

This literature review introduces concepts central to this thesis, including discussions on bird vocalisations, challenges in automatic monitoring, and technical insights into Few-Shot Learning and learnable frontends. The chapter is structured as follows: Section 2.1 provides an overview of bird vocalisation, encompassing its production, structure, and significance in communication. Additionally, a brief discussion is included on how the analysis of bird audio draws inspiration from speech analysis but bird audio differs fundamentally from human speech. Section 2.2 discusses hardware used in recording bird audio and the challenges faced while making field recordings. This section also incorporates explanations and reviews of the primary tasks in monitoring birds through their audio: activity detection and species identification. Section 2.3 offers a comprehensive review of FSL, detailing methods and various techniques, with a subsequent discussion on the application of FSL in audio. In Section 2.4, an in-depth review of learnable frontends is presented, covering learnable filterbanks and compression techniques. Finally, Section 2.5 provides a succinct overview of the datasets utilised in this work, while Section 2.6 discusses the relevance of the literature reviewed in this chapter to this thesis.

2.1 Bird Vocalisations

Bird vocalisation takes many forms, typically broken down into calls and songs [26]. Calls are concise and straightforward vocal expressions used by both male and female birds to convey various messages related to actions or circumstances, such as flight, threats, alarms, and territorial disputes. On the other hand, songs are typically characterised as more extensive and intricate vocalisations, primarily performed by male birds during the breeding season to attract potential mates. There are many exceptions to this, as numerous bird species, both male and female, produce complex, song-like vocalisations throughout the year. There is also overlap between calls and songs, and birds tend to be identifiable by both. In this work, we will refer to both bird calls and songs collectively as 'bird vocalisations' when discussing avian communication.

This section outlines the characteristics of bird vocalisations from a signal perspective, drawing parallels to human speech while highlighting their unique characteristics.
Section 2.1.1 provides a brief overview of the biological aspects of bird vocal production, offering a comparative analysis with human vocal production. Section 2.1.2 outlines the structure of bird calls and songs. In Section 2.1.3, the disparities between human speech signals and bird vocalisations are briefly explored. Finally, Section 2.1.4 briefly addresses the significance of vocalisation as a communication method for birds and its relevance to conservation efforts.

2.1.1 Production of Bird Vocalisations

The vocal system of birds is sufficiently similar to that of humans to justify comparisons between the two [11, 49]. The syrinx is the sound producing element in birds, functioning similarly to a human larynx, and contains membranes which vibrate when air from the lungs is forced over them. A primary difference between the two systems is location: whereas the human larynx is located on top of the trachea, a bird's syrinx is located at the bronchial junction (as illustrated in Figure 2.1). This leads to the interesting conclusion that birds have two potential sound sources, one at the top of each bronchus. This allows for complex, two-voiced song to be produced.

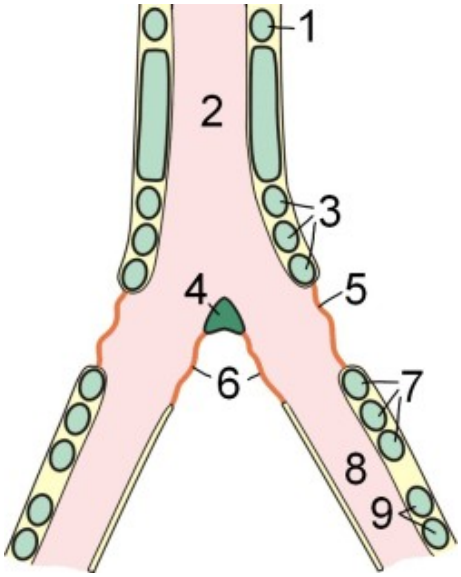

Figure 2.1: A Schematic drawing of an avian syrinx: (1)Cartilaginous Tracheal Ring, (2)Trachea, (3)First Group of Syringeal rings, (4)Pessulus, (5)Membrana Tympaniformis Lateralis (MTL), (6)Membrana Tympaniformis Medialis (MTM), (7)Second Group of Syringeal rings, (8)Main Bronchus, (9)Bronchial Cartilage. "Syrinx" by Uwe Gille, licensed under CC BY-SA 3.0.

Both the human larynx and the avian syrinx produce a harmonic signal, with strong energy at the fundamental frequency F_0, and relatively smaller amplitudes in the higher harmonics. This harmonic signal is produced by the vibrations of the Membrana Tympaniformis, both Lateralis (MTL) and Medialis (MTM) (designated as 5 and 6 in Figure 2.1), and the Pessulus (labelled as 4 in Figure 2.1). Modulations in the vocal tract, including those in the trachea as depicted in Figure 2.1, and within the mouth, influence the resulting vocalisations.

Similar to humans, birds employ their vocal tract and mouths to serve as filters and resonators, shaping the sound emanating from the syrinx before it is heard as a vocalisation [10]. The vocal tract may act as a lowpass filter, which combined with the higher F_0 of bird vocalisations and the consequent wider interval between harmonics, can result in pure tones in their vocalisations. Bird vocalisations may also include more harmonic vocalisations, polyphonic vocalisations (thanks to the two-voiced syrinx) or even non-periodic, unvoiced noisy sounds.

The beak also plays some role in sound production, facilitating the adjustment of the vocal

tract's length. Similar to humans, birds need to modify their vocal tract's length to produce higher or lower-pitched sounds. Additionally, to produce a high frequency sound, a bird must open their bill wide, whereas to produce a low frequency sound a bird must close their bill [25]. There exists a physical limitation on how rapidly a bird can open and close its bill, resulting in a trade-off between the bandwidth of a vocalisation and the rate at which it can be repeated. This trade-off sets a 'performance limit' on bird vocalisations, a species-dependent characteristic that follows a triangular distribution [145, 146].

These parallels between the human vocal system and avian vocal systems have generated considerable interest in bird audio research within the field of speech analysis. The first application of the source-filter model to birds was proposed by Nowicki in [132], and has been backed up by further research in [10, 95, 158]. The source-filter model is widely acknowledged as an appropriate framework for understanding sound production in birds (as well as humans). Consequently, many features developed for speech analysis have found extensive application in the analysis of bird vocalisations [46, 148].

2.1.2 Vocalisation Structure

Bird vocalisations, especially songs, share a linguistic structure akin to human speech, with hierarchical units such as elements, syllables, and phrases [193]. Elements are the basic building blocks of bird vocalisations, encompassing individual notes, chirps, harmonic sounds or whistles, which combine to form more complex structures. Syllables, comprised of one or more elements in a specific sequence, amalgamate to create a phrase, an extended vocalisation conveying a specific message. Phrases, in turn, can be strung together to compose a song [26]. Figure 2.2 illustrates how the song of a Eurasian Wren can be deconstructed into these units, highlighting elements, syllables, and phrases.

Moreover, birds often possess a repertoire of several songs, each characterised by the order, types, and number of phrases employed, contributing to the diversity of a bird's vocal expressions [27]. Repertoire size varies by species, while some birds may only have one song and a few calls, species such as the brown thrasher are known for their repertoire of over 1000 songs. Whilst the size of a species repertoire may pose issues in generalising recognition

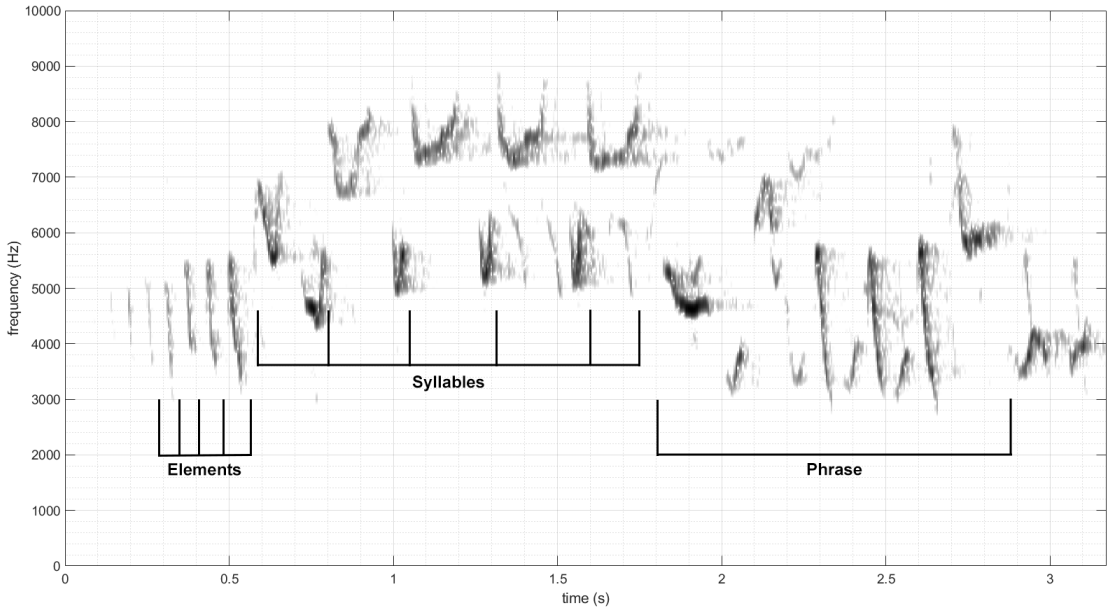

Figure 2.2: Spectrogram Representation of Eurasian Wren song, labelling elements, syllables and phrases.

of that species, it is worth noting that even the most complex songs are constructed out of the limited set of syllables and phrases available to a bird, with songs being complex combinations of these elements.

Distinct from songs, bird calls are shorter and less complex vocalisations, contextualised for specific situations such as alarm, flight or feeding [26]. While calls may be constructed from the same linguistic units as songs, they serve specific purposes within their context, communicating information related to actions or signalling territorial disputes to fellow members of the population. Calls contain both voiced and unvoiced sounds, whereas songs predominantly consist of voiced elements.

2.1.3 Difference from Human Speech

Despite the similarities in production between bird vocalisations and human speech, there are significant differences between the two signals, both acoustically and linguistically. Although this may be stating the obvious, it warrants a brief discussion, given the implications of these differences on the design of systems utilising these signals. Human speech is characterised by

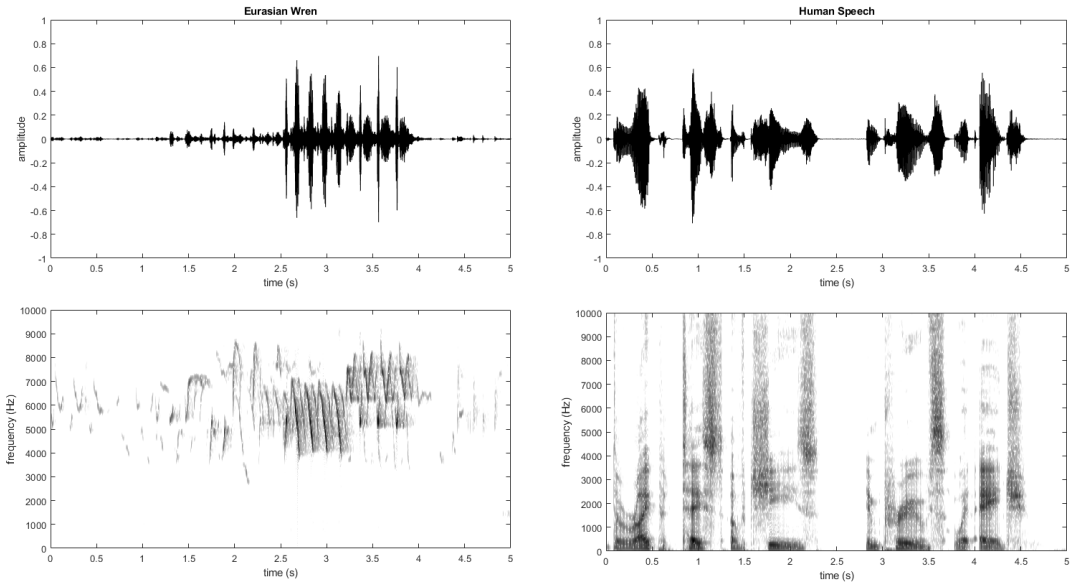

Figure 2.3: Temporal and Spectrogram representations of clean bird vocalisations (Eurasian Wren) and clean Human Speech. Spectrograms are calculated using the same window sizes (25 ms) and overlap (50%).

greater harmonic richness, longer and more prolonged syllables/phonemes, significantly lower pitch, and the absence of trill rate and frequency modulation features prominent in bird vocalisations. Figure 2.3 provides a direct comparison between a clean example of human speech and a bird call, both normalised and aligned on the same time/frequency scale. We can see that human speech does indeed differ from bird vocalisations in the manner discussed above.

The perception of each species vocalisations is also important to note. The mel frequency scale, based on human audio perception, is well-suited for analysing music and voice intended for human ears. However, bird vocalisations, although audible to humans within our hearing range (on average, birds hear best between 1 and 5 kHz, with exceptions [47]), are not intended for human reception.

It is with these differences in mind that we feel that perceptual scales, such as the mel scale and features derived from these scales, such as MFCC are not optimal for bird audio [65]. While Mel-based features have proven their worth and shown promising results in various

studies, investigating features more tailored to the unique characteristics of bird vocalisations may offer a more effective approach.

2.1.4 Importance of Vocalisation

Bird vocalisations play a crucial role in facilitating communication among members of the same species, different species, and are important to conservation and monitoring efforts by humans. Birds vocalise to communicate with members of their own species and can discern neighbours from both their own and other species through their calls and songs [120]. These vocalisations convey information about species, sex, actions, threats, and sources of food.

Vocalisations serve as an incredibly efficient means of communication for birds. Vocalisations can be heard day or night, and require less energy than physical movement. They are especially effective in environments with limited visibility, such as canopies or forests, providing safety for birds while maintaining communication. Songs, in particular, play a pivotal role in species recognition and mate selection [26]. Some argue that differences in songs between bird populations can serve as a more reliable indicator of species distinction than morphological characteristics [133, 156, 197].

In terms of monitoring bird populations for ecological, zoological, and conservation purposes, the effectiveness of vocalisations also carries over. Collecting vocalisation data offers practical advantages over methods like catch-and-release, as sound recording is more straightforward and cost-effective, enabling the collection of large datasets [39]. Moreover, given that birds are often concealed or in low-visibility environments and may be disturbed by human presence, audio data becomes a valuable tool for monitoring and identification, even when visual identification is challenging. The importance and effectiveness of bird vocalisations in their communication, coupled with their applicability in monitoring scenarios, drives the motivation to monitor bird populations through their vocalisations.

2.2 Automatic Monitoring of Birds

The use of Automatic Recording Units (ARU) in ecological studies has increased [39, 45, 188]. Their advantages include minimising subject disturbance and temporal bias, recording

over extended periods, and deployment in challenging environments where field recording may pose difficulties [121]. These attributes make them an attractive choice for ecological studies on birds. It is worth providing context into the type of hardware that is used for recording of bird vocalisations, the expected quality of recordings, and the environments and methods used. Automatic systems capable of continuous operation in remote areas of interest are a crucial initial step in species identification and population monitoring tasks [154].

This section also outlines the two primary tasks in monitoring bird populations through vocalisations. Section 2.2.1 briefly outlines the type of hardware used and the capabilities of ARUs, along with challenges in monitoring bird populations using audio. Section 2.2.2 gives insight into the two major tasks involved in monitoring bird populations: activity detection and species identification.

2.2.1 Recording Hardware & Challenges

Recording hardware typically comprises a recording unit and one or more microphone capsules. Although they do not perform any classification tasks, they can be programmed to record during specific periods (such as dawn or evening chorus), under certain conditions (e.g., sound level or SNR above a specified threshold), and to exclude recordings lacking sufficient energy in a specific frequency band. ARUs such as the Wildlife Acoustics SM4[1] and the Cornell SwiftOne[2] (pictured in Figure 2.4) are equipped with such features, along with the ability to export metadata, including latitude and longitude. They are typically designed for multi-channel audio recording, utilising sensitive microphones that can be deployed away from the unit via cables. As an improvement on the scrubbing features already found on these units, bird activity detection methods could further decrease the amount of data stored, reducing it to only data that requires subsequent processing.

Many examples of bird vocalisations used for bird species research have been contributed by the public to repositories like Xeno-Canto[3], utilising equipment of varying quality. These recordings constitute a portion of many datasets, ranging from those captured on

[1] https://www.wildlifeacoustics.com/products/song-meter-sm4
[2] https://www.birds.cornell.edu/ccb/swift-one/
[3] https://xeno-canto.org/

Figure 2.4: Two examples of Automatic Recording Units: the Wildlife Acoustics SM4 (left) and the Cornell SwiftOne (right). Images sourced from the respective product pages of each unit.

smartphones with limited dynamic range and sensitivity, to those recorded with professional-grade audio equipment. This variability in recording hardware, affecting everything from sensitivity to noise level to frequency response and directionality, poses challenges to monitoring birds through their audio.

Several challenges accompany the acoustic monitoring of bird populations, briefly outlined here. Recordings obtained in areas with diverse sources of non-stationary noise present difficulties in consistent analysis and species recognition, as eliminating noise sources without compromising bird audio proves challenging (insects, wind, and non-target species add to the complexity).

Due to sound wave attenuation with distance, particularly for higher frequencies emitted by birds, and additional attenuation from the environment, capturing weak signals from target species becomes challenging. These far-field recordings have posed issues for certain detection methods.

While proper surveying techniques fall under the purview of ornithologists and ecology researchers, understanding these challenges is crucial for effectively processing bird audio. The quality of raw data from field recordings significantly influences the efficacy of recognition methods.

2.2.2 Tasks in Bird Audio Monitoring/Bioacoustics

Research into the automated recognition of bird vocalisations has paralleled developments in several speech tasks such as automatic speech recognition and speaker verification [148]. This arises from the similarities in signal production and the applicability of the source-filter model, as discussed in Section 2.1.1.

The predominant tasks in this domain are Species Recognition/Identification (along with the related problem of Species Diarisation[4]) and Species Agnostic Bird Activity Detection (BAD). Species recognition entails identifying an utterance as belonging to a specific species within a closed set of candidate species. On the other hand, BAD focuses on identifying the start and end times of a bird vocalisation, irrespective of species or call/song type. Whilst the end result of BAD may seem less specific than species recognition, it addresses a more general problem and serves as a crucial filtering step for obtaining data in species recognition.

This thesis predominantly centres on BAD, featured prominently in Chapters 3, 4 and 6. However, species identification also appears in Chapters 5 and 6. Given the substantial overlap between these two challenges, a review of literature covering both problems is warranted. Notably, in 2016, the organisers of the BirdCLEF challenge declared the "arrival of deep learning" for bird audio tasks [62]. Consequently, this review predominantly concentrates on contributions to the field post-2016.

Activity Detection

Species agnostic BAD poses the problem of identifying bird activity without incorporating any prior knowledge of species. It serves as an important initial step in filtering relevant data from audio collected during monitoring operations, reducing manual checking of audio and spectrograms by researchers. In a way, it aims to identify segments of audio that contain the 'characteristic melody' of bird vocalisations. Species/vocalisation agnostic activity detection is a much more recent problem than the species identification task. It should be noted that an activity detector can also be species specific, detecting only the species of interest [125].

While the output of such a recogniser is less specific than species recognition (a bird of

[4]The problem of 'Who sang when?'

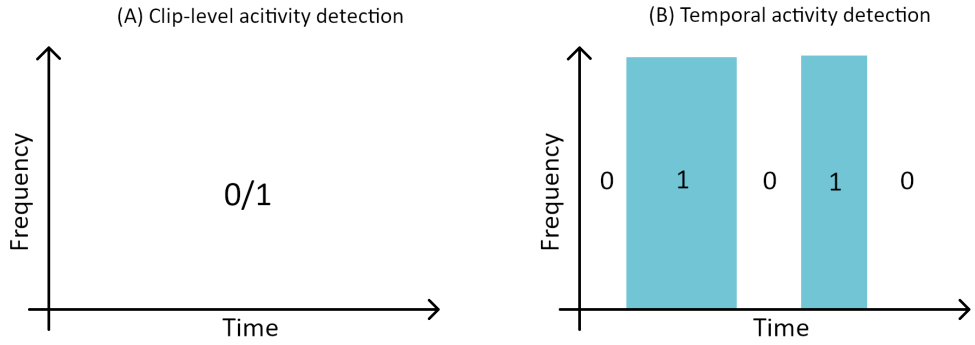

Figure 2.5: Activity detection can be clip-level (A), or temporal (B). Clip-level annotation discerns the presence or absence of bird vocalisations in a clip of audio, with no consideration as to where or for how long. Temporal activity detection provides information of event onset and offset times.

species X is vocalising) or diarisation (bird X is vocalising at time Y for duration Z), it requires generalisation across many species and vocalisation types, adding complexity due to environmental factors and noise. Bird vocalisations vary by species, sub-species and even between different populations [70, 133] and as discussed in Section 2.1, vocalisation types vary, making generalisation challenging.

BAD can be further categorised into two types of activity detection: clip-level and temporal. The difference between these types is illustrated in Figure 2.5. Clip-level activity detection (Figure 2.5(A)) is a basic form that discerns the presence or absence of birds in a sound clip, providing a binary output. This is useful to detect whether a bird is present in a recording, or segment of a recording, and can be used as the first step in further segmentation or processing. Although useful for detecting bird presence, it lacks temporal resolution and merges multiple events together. Temporal activity detection (Figure 2.5(B)), akin to Voice Activity Detection in speech processing [100, 178], outputs the onset and offsets of events, segmenting the audio into regions with events of interest. This type of activity detection is more difficult, employing less data to make decisions and requiring consideration of onset and offset time calculations. In the monophonic case, overlapping events are merged, but temporal activity detection can extend to the polyphonic case, where multiple concurrent events have their own onset and offset boundaries.

Energy-based detection, as employed by some studies [183], proves effective primarily in low-noise recordings. However, in recordings with higher levels of noise, energy-based BAD lacks robustness, necessitating the use of more advanced methods. Stowell et al. have proposed challenges [183, 187] in recent years, specifically addressing the issue of BAD. Many successful entries in these challenges leverage Convolutional Neural Networks (CNN) in their detection systems.

Grill & Schlüter submitted the best performing system in the 2016/2017 challenge with their system 'bulbul' [67], which used a wide receptive field (the entire audio clip, 10 s) to determine whether there was bird activity. The 'bulbul' system achieved an Area Under Receiver Operating Curve (AUC) of 0.899. They also proposed a local architecture to counteract potential problems of differing temporal positions of calls, however this system performed no better. The 'bulbul' system was later adapted and served as a baseline in the 2018 DCASE BAD challenge. Cakir et al. [23] explored the temporal context of bird vocalisations using a Convolutional Recurrent Neural Networks (CRNN). The authors also introduced separation of max-pooling into frequency-based pooling in the feature extractor and temporal pooling after the recurrent layers. Incorporating this temporal context, the system achieved results comparable to 'bulbul', with an AUC of 0.895.

In the 2018 challenge, Liaqat et al. [104] adopted an architecture similar to Grill & Schlüter's 'bulbul' [67], incorporating data augmentation and domain adaptation. Their findings provided some evidence that resolution in the temporal domain, is more important than high resolution in the frequency domain. Liaqat et al.'s submission achieved an AUC of 0.839, but failed to outperform the 'bulbul' baseline, which achieved an AUC 0.885 on this data. Mario Lasseck's system [98] utilised deeper networks, data augmentation, and ensemble modelling to outperform the 'bulbul' baseline, albeit with increased model size and complexity. This system was the top performing system in the 2018 challenge, achieving an AUC of 0.890. Across both challenges, systems often utilised log mel-spectrograms as input, and those not employing deep learning were outperformed by their deep learning counterparts.

In 2019, Lostanlen et al. [117] demonstrated improved performance in a BAD task using single-channel audio through the application of Per-Channel Energy Normalisation (PCEN).

This finding is reproduced in our experiments assessing popular acoustic frontends on bird audio (Section 5.3.4), where PCEN significantly enhances performance compared to log mel-spectrograms, with a relative improvement of 25% in AUC. Section 2.4.3 provides a discussion of PCEN as a learnable frontend, with a detailed explanation available in Section 5.2.3. The authors in [117] also show that the incorporation of long-term time-frequency summary statistics increases performance in the multi-channel or multi-sensor case.

In 2018, Balestriero et al. [7] expanded on the work of Grill & Schlüter [67] by introducing a learnable time-domain filterbank layer to the network. Using a spline-interpolation-based filterbank, they learned the basis for a Time-Frequency (TF)-representation with constraints on smoothness and boundary conditions. Their results demonstrated that this learnable filterbank outperformed a mel-spectrogram baseline, with a modest relative enhancement of 2% in the AUC metric. In 2021, Zeghidour et al. [216] introduced Learnable Audio Frontend (LEAF), further discussed in Section 2.4.2 and Section 5.2.4. As part of the author's evaluation of LEAF, they tested it on a BAD task using EfficientNet-B0 [190], revealing that the usage of LEAF improved performance over log mel-spectrograms. Both studies [7, 216] show that mel-derived features may not be the most suitable choice for bird, or more broadly, bioacoustic audio, advocating for learning TF-representations directly from the data.

In recent years, FSL-based activity detection has gained popularity due to challenges proposed by the DCASE community [125, 130]. This approach involves detecting activity using only a few labelled examples of the target class. Research into FSL-based approaches is an active area of research for both BAD and bioacoustic activity detection more generally. While briefly mentioned here as part of the broader literature on BAD, a more detailed discussion of FSL for bioacoustics can be found in Section 2.3.2.

Species Identification/Recognition

The objective of Bird Species Identification (BSID) is to identify a bird species from a sample of their vocalisation. While there is some overlap between the two tasks, BSID is more specific than a general BAD task, leveraging the unique qualities of one or more species'

songs to train specific models. BSID makes use of particular species characteristics in vocalisation such as frequency range, trill rate, syllable and phrase structure etc.. Instead of a characteristic melody of bird vocalisations, the specific features of each species vocalisation is included in the model.

BSID is typically a multiclass task where the aim is to identify an audio clip as belonging to one species in a set. It usually operates under the assumption that the audio in question contains a bird vocalisation for one of the target species, necessitating BAD as an initial filtering step. The combination of both tasks is bird audio diarisation [173]. However, some systems may output an 'unknown' class, referring to either unknown species or no activity.

BSID presents its own set of challenges. One is the variability in song between different populations [70, 133]. Additionally, environmental factors pose an issue. Several studies have shown that birds engage in a spectrum sharing practice, depending on their environment and the presence of other species [68, 174, 175]. They also adjust their amplitude to suit their surroundings [20], potentially leading to changes in other aspects of the vocalisation due to the Lombard effect [21]. Such changes may incorrectly identify a bird as not present, or as another species.

The trajectory of BSID is closely related to research in Automatic Speech Recognition systems, and has also taken inspiration from musical instrument and genre classification tasks. Early approaches have employed the usage of Dynamic Time Warping with template matching, and Hidden Markov Models (HMM) utilising features such as MFCC or Linear Predictive Coding Coefficients [148]. However, these methods are susceptible to noise issues.

Other HMM-based approaches have made use of spectral peaks and frequency tracks to construct models to recognise several species. Jančovič and Köküer have proposed several, constantly improving systems using this method as a foundation [78, 79, 80], each time increasing the amount of bird species that could be recognised successfully. Stastny et al. [179] introduced a system recognising 18 species using Human Factor Cepstral Coefficients. Stowell et al. [182] have also used HMMs to analyse the temporal structure of songs within species.

Less commonly used classification methods include Support Vector Machines (SVM) and

Gaussian Mixture Models (GMM). Fagerlund used SVMs in conjunction with decision trees, achieving good accuracy within a limited species range [52]. GMMs have been used by Fagerlund et al. in [53] using MFCC based features to identify 14 bird species. In recent years, CNNs have gained popularity in BSID systems, demonstrating state-of-the-art performance with the increasing adoption of deep learning approaches [62].

The BirdCLEF challenges, held annually since 2014, have played a crucial role in driving BSID research. The 2021 edition [84] focused on detecting and identifying bird vocalisations from 397 species in continuous soundscapes. This challenge provided one of the largest fully annotated collections of bird audio, encompassing a diverse range of species. Top submissions utilised pretrained CNN architectures fine-tuned on the data, coupled with post-processing of model output, with the top submission achieving an F1-score of 0.6932 on the main task.

In the 2022 [86] edition, the number of bird species was reduced to 152, and the focus shifted towards analysing rare and endangered species with limited labelled data. This edition aimed for a Few-Shot Learning structure, although many teams addressed data scarcity by employing extensive data augmentation. The top submission in 2022 achieved an F1-score of 0.8527.

The most recent 2023 edition [85] continued the theme of including species with few labelled recordings but increased the number of species to over 1000. Notably, this challenge imposed a computational limit on inference to encourage the efficient use of computational resources on modest hardware available to conservationists. Consequently, the 2023 challenge witnessed extensive optimisation efforts, aligning with the goal of reducing resource usage for large-scale, widely deployed computation in ecology and environmental applications [184]. The top submission of BirdCLEF 2023 achieved a class-wise mean average precision (cmAP) of 0.844.

In 2021, Kahl et al. [87] introduced BirdNET, currently the leading BSID model due to continuous support and development. BirdNET is not only a sophisticated BSID system but also a citizen science initiative with a smartphone app available on major platforms. This unique approach has enabled BirdNET to evolve, recognising approximately 3000 of the world's most common bird species in 2023, up from 984 species in 2021. The architecture of

BirdNET is based on ResNet [71], incorporating knowledge distillation, data augmentation, and high temporal resolution in spectrogram input (based on findings in [104]). As of the 2022 publication, BirdNET achieves an accuracy of 0.777 and an AUC of 0.974 on focal recordings.

The success of BirdNET can be attributed to its extensive and curated training data, which includes publicly available bird vocalisation datasets and datasets containing non-bird signals to create a 'non-event' class. While this large data approach suits the citizen science context, it may not adequately cover rare and underrepresented species, necessitating further expert annotation.

Polyphonic BSID addresses scenarios with multiple, overlapping events, presenting a more challenging task than monophonic or non-overlapping BSID. As many birds sing together in choruses, and studies often use omni-directional instead of focal recordings, polyphonic BSID becomes a natural progression as monophonic BSID performance improves. BirdCLEF 2021 [84] focused on polyphonic soundscape analysis, with the top entry achieving an F1-Score of 0.6932 (baseline for the challenge was 0.4799 when all segments were marked as non-events). BirdNET [87] has also been evaluated on a polyphonic soundscape task, exhibiting drastically decreased performance from its monophonic, focal recording performance, with AUC dropping from 0.974 to 0.596.

In 2022, Parilla and Stowell [140] trained a CRNN using progressively more overlapping data, eventually handling 10 overlapping events. Results suggest that training with a similar amount of polyphony as expected during inference yields optimal performance, whereas training with too many overlapping events degrades performance on audio with low polyphony. While polyphonic BSID and soundscape analysis remains challenging, training models from low to high polyphony and further advancements in monophonic BSID contribute to improvements in polyphonic BSID.

2.3 Few-Shot Learning

Few-Shot Learning (FSL) is a machine learning approach that trains models to generalise effectively and make accurate predictions when provided with limited labelled data. In a FSL

scenario, a small labelled *support set* (denoted as S) is provided, and the objective is to classify the elements of the *query set* Q. This support set comprises N labelled examples, or *shots*, for each class.

Lake et al. [97] state that a worthy goal would be to build an AI system that performs better than a human, using the same amount and kind of training that a human may receive. FSL draws inspiration from human learning and generalisation abilities, where humans can quickly grasp new concepts, visually recognise objects, or identify audio with just a few examples. FSL aims to equip machine learning models with similar capabilities by training them to discern common patterns and relationships from a limited dataset, mimicking how humans leverage prior knowledge and experience. This is particularly valuable for tasks where data is scarce [209].

Wang et al. [207] provide a more precise definition of FSL, framing it as a machine learning problem involving a task (e.g., image classification), an experience (labelled images), and a performance metric. Notably, the experience contains only a limited number of examples for the target task. The authors emphasise that the main challenge of FSL is in its potential to act as an unreliable empirical risk minimiser [207]. This means that despite minimising training data loss, a poorly designed FSL system might struggle to generalise to new, unseen data due to overfitting issues arising from limited training data.

In the early days of FSL, one-shot learning (a specific case of FSL using only one example) was explored by Fink [56] in 2004, and Fei-Fei et al. [54] in 2006. These studies focused on object classification in images and introduced FSL as a machine learning paradigm. Notably, these works predate the rise of deep learning systems and relied on less computationally intensive machine learning methods, such as nearest neighbours classification [56] or Bayesian modelling [54]. Fink [56] framed the learning problem as one of discovering a distance function, where the distance between two instances of the same class is smaller than the distance between any two instances from different classes. This is an example of a meta-learning (i.e. 'learning to learn') strategy that remains popular and effective in FSL [118, 139, 207].

Additional approaches to the FSL problem have emerged in the following two decades.

Parnami et al. [139] offer a useful taxonomy of these approaches, providing an overview of systems within each category. They categorise FSL systems into two main groups: those based on meta-learning and those not based on meta-learning. Meta-learning based approaches are further broken down by what aspect of the model is subject to meta-learning. This taxonomy, as well as the approaches themselves will be discussed in more detail below in Section 2.3.1.

Wang et al. [207] propose a broader taxonomy of FSL that focuses on how each approach impacts the hypothesis space. They classify FSL systems into three groups: data-based systems that incorporate prior knowledge to enhance training data; model-based systems that use prior knowledge to limit the hypothesis space; and algorithm-based systems that integrate prior knowledge into the parameter search and optimisation strategy. It is important to note that some overlap exists between these two taxonomies, as meta-learning can be found in both the model and algorithm categories outlined in [207].

2.3.1 Approaches to Few-Shot Learning

As mentioned earlier, Parnami et al. [139] provide a useful taxonomy of FSL approaches, visualised in Figure 2.6. Non-meta-learning approaches include transfer learning-based methods, which are a simple and sometimes effective approach [195, 206], and transductive methods (some of which will be elaborated on below). Meta-learning approaches are further subdivided into metric-based, optimisation-based and model-based meta-learners:

- **Metric-based**: High dimensional data is transformed to a lower dimensional space using some embedding function f_θ. Metric-based meta-learning is tasked with either learning the embedding function f_θ given a distance function d (such as Euclidean distance), or learning both the embedding function f_{θ_1} and distance function d_{θ_2} jointly. Metric based meta-learners are easily deployed, easily understood and have fast inference. However, they tend to be less adaptive than other approaches. They are well suited for use with simple, non parametric classifiers (such as nearest neighbours).
- **Optimisation-based**: Traditional gradient-based optimisation in machine learning is not designed for use with few data, and will usually lead to overfitting.

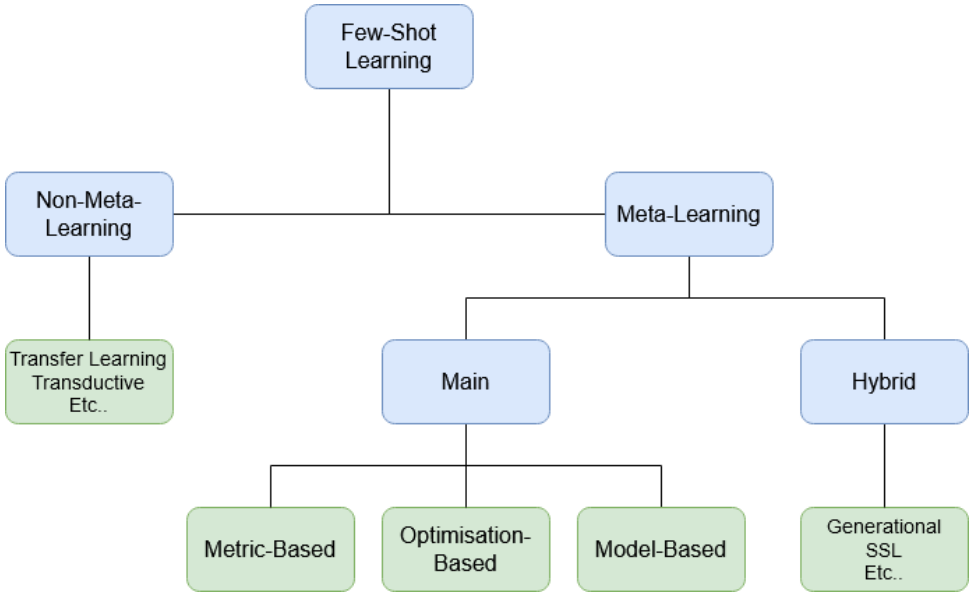

Figure 2.6: Simplified categorisation of FSL approaches by Parnami et al. [139]. The authors categorise approaches to FSL as meta-learning-based FSL and non-meta-learning-based FSL. Meta-learning-based approaches are further broken down into metric, optimisation and model-based methods. Approaches using a combination of meta-learning and other techniques are categorised as 'hybrid' approaches.

Optimisation-based meta-learning methods for FSL involve the design of optimisation strategies that can achieve good generalisation using a small number of support samples. This typically involves learning parameters for the optimiser which facilitate quick and easy fine-tuning. Optimisation-based methods tend to be more adaptive than metric-based or model-based methods, however they require additional gradient-optimisation steps at inference time and can still overfit to the limited data.

- **Model-based**: Model-based meta-learning is the least commonly employed approach. Model-based meta-learning involves the creation of architectures designed to learn and adapt quickly. This is achieved in myriad ways, such as the usage of external memory which the model has access to allowing it to store and read information, to networks using 'fast-weights' where another neural network is designed to predict the weights of the model. Model-based approaches eliminate the need to define metrics, or to optimise further at inference time, however they are difficult to design and less parameter efficient than their metric and optimisation-based counterparts.

Parnami et al. [139] also give examples of hybrid meta-learning approaches, which include some element of meta-learning alongside other techniques such as generating additional data to supplement the support set [205] or self-supervised learning techniques [30]. However most meta-learning approaches to FSL fall into the three categories above, with the majority being either metric or optimisation-based methods.

As previously mentioned, a subset of non-meta-learning-based FSL approaches known as transductive inference learning methods, are also popular. Research from Tian et al. [195] in 2020 even suggests that a well-learned embedding model for FSL can be acquired without resorting to meta-learning. Instead, leveraging techniques such as knowledge distillation and data augmentation. However, it is essential to note that meta-learning methods find use in the majority of FSL systems [139].

Below, an overview of various FSL systems is provided. The meta-learning based approaches will be introduced first, starting with various metric-based meta-learners and their developments. Following this, we will delve into descriptions of optimiser-based meta-learners. Finally, the discussion will shift to non-meta-learning-based approaches, covering techniques like fine-tuning/transfer learning and transductive inference-based methods.

Meta-Learning Based Approaches

After early endeavours [54, 56, 96], which initially formulated the concept of one-shot learning and by extension FSL, the topic went through a quiet phase until the emergence of deep learning architectures and the growing demand for large amounts of labelled data. The first to incorporate deep neural networks, specifically CNN architectures, were Siamese Neural Networks, introduced by Koch et al. [92] in 2015.

Siamese Neural Networks, an early example of metric-based meta-learning using deep learning, feature two identical sub-networks sharing the same parameters (referred to as twins). When provided with a pair of images, one for each network, their goal is to determine the degree of similarity or dissimilarity between these images. Each network generates an embedding, and a learnable distance function quantifies the dissimilarity between these

embeddings. For classification, a softmax function is applied to the distances between the query image's embedding and the support images' embeddings.

Matching Networks, proposed by Vinyals et al. [203] in 2016, introduce the idea of episodic learning to FSL, mirroring the inference-time scenario during training. This means each training step is formulated as a N-shot K-way task, where N matches the number of 'shots' at inference time. A more detailed explanation of episodic learning is provided in Section 4.1.2. These are also metric-based meta-learners, but focus on learning the embedding function and utilise a fixed distance function, namely cosine distance, to measure the query data's similarity to support sets. They employ an attention mechanism to recall relevant support set features and use the complete support set embeddings when constructing each query embedding, achieved through an LSTM-based model.

In 2017, Snell et al. proposed Prototypical Networks [177], building upon the episodic training training method introduced in [203]. Prototypical networks, often referred to as protonets, operate on the principle that there is a mapping from the feature space to an embedding space, where data points belonging to the same class tend to cluster around a single prototypical representation of that class. In practice, this prototype embedding is calculated as the mean of the support set embeddings. These characteristics make protonets metric-based meta-learners, using a fixed distance function.

The authors make a compelling case for using Bregman Divergences [18] as the distance functions used for protonets. This is empirically supported by comparing protonets trained with cosine distance (as in Matching Networks [203]) and squared Euclidean distance (a Bregman divergence, as discussed in [9] and Section 4.1.1). A more detailed explanation of how protonets are implemented, and why Bregman divergences are suitable choices in distance function can be seen in Section 4.1.1.

Protonets are appealing in the context of FSL due to their straightforward mechanisms and ease of training. They surpass Matching Networks in performance across various tasks and have become one of the most widely used FSL frameworks. An enhanced variation of protonets utilising Semi-Supervised Learning was proposed by Ren et al. [155] in 2018. Although this approach improves upon the results of the original protonet [177], it introduces

greater complexity into the training process, necessitating additional refinement.

Oreshkin et al. [136] argue that the performance gap between cosine distance and squared Euclidean distance in [177] results from the scale of these distances and its impact on the cost function (negative log probability via softmax over distances). They propose leveraging the relationship between the distance function and cost function, introducing metric scaling to enhance metric-based meta-learners' performance. Metric scaling was shown to effectively close the performance gap between cosine and squared Euclidean distances, but it necessitates fine-tuning using cross-validation techniques, increasing development time.

Relation Networks, introduced by Sung et al. [189] in 2018, share similarities with protonets. Relation Networks utilise episodic learning [203] and prototype representations [177] from Matching Networks and Prototypical Networks, respectively. However, what sets Relation Networks apart is the trainable network acting as the distance function (referred to as the relation module). This feature harkens back to earlier FSL attempts [56, 92]. Despite the inclusion of a learned distance function, Relation Networks offer only slight performance improvements over Prototypical Networks when a higher number of shots are used. However, in comparative studies, Relation Networks have shown the capability to outperform Prototypical Networks when a shallower backend model is employed [31].

More recently, metric-learning approaches have incorporated parametric classifiers and transfer learning. Meta-Opt, proposed by Lee et al. [99] in 2019, incorporates non-linear, parametric classifiers (previously not utilised to focus on learning a good embedding function from few data). This is achieved by introducing a differentiable quadratic programming solver to the network, enabling the optimisation of of SVMs with few data. Other systems such as Meta-Baseline, proposed by Chen et al. [32] in 2021, utilise a two-stage training approach. The first stage involves supervised classification, trained conventionally, followed by a meta-learning refinement step to adapt the embedding function for FSL applications. Meta-Baseline also combines elements such as episodic learning [203], prototype representation [177] and metric scaling [136].

The systems mentioned above are primarily metric-based meta-learners, but several optimisation-based meta-learners have also demonstrated state-of-the-art results. For

instance, Meta-LSTM [152] (proposed by Ravi et al. 2016) and Meta-SGD [103] (by Li et al. in 2017) focus on meta-learning model initialisations and optimiser parameters that enable rapid generalisation from limited data. Both methods aim to maximise the meta-learner's ability to generalise across the tasks encountered during training. As their names suggest, Meta-SGD achieves this through a modification of the Stochastic Gradient Descent (SGD) algorithm, targeting parameters that allow SGD to adapt the model to new tasks with minimal data. On the other hand, Meta-LSTM employs an LSTM network to learn a learning rate schedule for the same purpose. Although both methods outperformed Matching Networks in an image classification task, they were surpassed by Prototypical Networks [177], which were also introduced in 2017.

Among optimiser-based meta-learning approaches, the most influential system is MAML [57]. MAML, introduced by Finn et al. in 2017, is a framework which encourages learning a model applicable to a wide range of tasks during training and requires only a small number of update steps at inference time. MAML optimises over all tasks in a distribution during training to learn a generalised model, which is then specialised for a specific task during fine-tuning, using the support set relevant to that task. MAML has been found to achieve better performance than its contemporary, Protoypical Networks, however it does not adapt as well to domain shifts [31].

This overview of meta-learning-based approaches to FSL has primarily concentrated on metric-based approaches. While optimiser-based approaches show potential, the necessity for retraining and adaptation during inference makes them less appealing compared to metric-based learners, which require no gradient updates at inference time. Despite metric-based learners' limited adaptability, it is precisely this lack of adaptation and their ease of implementation that makes them an interesting prospect.

Non-meta-learning Based approaches

Meta-learning based approaches do not encompass the entire landscape of FSL. We will now look at some non-meta-learning-based FSL systems. Although these have been less prevalent in the literature, they are gaining popularity as the role of meta-learning in FSL is being

re-evaluated [94, 195]. The two most utilised non-meta-learning based approaches are embedding models trained in a standard way using cross-entropy, adapted for use with FSL and transductive inference.

In 2019, Wang et al. [206] introduced SimpleShot, a straightforward FSL method designed to serve as a reproducible baseline for comparison with other approaches. It employs an SGD-trained network to minimise the cross-entropy loss across all training set classes. The classifier layers are then omitted, leaving only the embedding function. Few-shot inference is conducted similarly to Prototypical Networks [177], where a prototype embedding is computed as the mean of support set embeddings, and a query point is assigned to the class with the closest prototype based on a distance function. SimpleShot also utilises data augmentation and feature transformations such as centring and L2-normalisation. SimpleShot achieves competitive results, especially when using larger backbones, sometimes even surpassing other methods mentioned above.

The Baseline and Baseline++ methods, introduced by Chen et al. [31] in 2019 and influenced by Dhillon et al.'s work [44], follow a similar approach. They involve training a new classifier using the support set of each class while keeping the embedding function fixed. Baseline employs a linear classifier, while Baseline++ utilises a cosine distance-based classifier. These networks, particularly Baseline++, seem to outperform many meta-learning-based approaches, especially when using larger backbones similar to SimpleShot [206].

Similar results emerge from the work of Tian et al. [195], who advocate for training the embedding function using a standard classification task with cross-entropy, avoiding meta-learning entirely. In contrast to SimpleShot [206], which utilises feature transformations to achieve performance similar to meta-learning approaches, Tian et al. utilise self-distillation alongside data augmentation to enhance performance. When using the same backbone network as popular meta-learning methods, their networks achieve comparable performance levels. This demonstrates that when combined with a sufficiently large training set and additional techniques, training embedding functions using cross-entropy, without meta-learning, can produce competitive results.

Another class of methods that typically avoids the use of meta-learning is transductive

inference. Transductive methods avoid learning a general model, common in meta-learning setups. Instead, they rely solely on the available data, which can include both labelled data (the support set) and unlabelled data (the query set, but also possibly additional data) to classify query points. These transductive methods can complement meta-learning based approaches [109], but they can also be used in conjunction with simpler cross-entropy-based methods outlined earlier [159], eliminating the need for meta-learning. However, transductive methods inherently require additional inference time adjustments, akin to optimisation-based meta-learning, and are slower compared to metric-based meta-learning methods.

The first to use transductive inference for FSL was Liu et al. [109], through their introduction of label propagation. In this approach, the authors combine a popular inductive meta-learning-based method, Prototypical Networks [177], with their transductive label propagation. Label propagation leverages both labelled and unlabelled data, utilising a graph-based approach to assign or propagate labels to unlabelled instances based on their neighbours. This propagation is performed iteratively until convergence, and the propagated labels are employed for classifying the query points. Embedding propagation, proposed by Rodriguez et al. [159] in 2020, extends this concept by also propagating the embeddings in a similar manner, followed by label propagation. Both approaches show marginally higher 5-shot classification performance than inductive methods, and substantially better one-shot performance.

In 2020, Boudiaf et al. [17] introduced Transductive Information Maximization (TIM), a method that maximises the mutual information between query features and their predicted labels while minimising the cross-entropy loss on the support set, akin to fine-tuning methods. This approach has further refinements [16, 38, 202] that address class imbalances during inference using the Kullback-Liebler Divergence. These systems do not rely on meta-learning or episodic training methods; instead, they involve fine-tuning steps applied to embedding functions trained using standard cross-entropy loss. The performance of TIM and its successors surpasses that of the aforementioned meta-learning-based approaches, as well as the simple fine-tuning-based methods.

To summarise, systems implementing a simple baseline trained using cross-entropy loss can

match the performance of meta-learning-based approaches when provided with sufficient data. Transductive methods enhance this further, leveraging both the support set (labelled data) and query set (unlabelled data) to improve classification performance and surpass meta-learning methods. However, they require additional inference time updates and the availability of all data at inference time, which may not be suitable for real-time applications.

All the systems described above have all been evaluated using image classification tasks and datasets. In Section 2.3.2, we discuss FSL and its application in audio, particularly in bioacoustics.

2.3.2 Few-Shot Learning with Audio

FSL with audio introduces distinct challenges compared to image classification. While the Short-Time Fourier Transform (STFT) can convert an audio channel from a 1-dimensional time series into a 2-dimensional spectrogram akin to an image, spectrograms cannot be treated identically to images. Spectrograms cannot be subject to arbitrary transformations such as cropping, rotation, and scaling, as any changes in the horizontal or vertical dimension directly affect the time and frequency content of the signal. Consequently, methods relying heavily on data augmentation [195, 205, 206] may not be suitable. There are also the issues of signal-to-noise ratio, and polyphonic, multi-label data [208], which may influence how the FSL task is set up.

Moreover, FSL for audio presents a different perspective on the concept of 'shots'. In image classification, the number of shots pertains to the images in the support set, all sharing the same size and dimensions. In the context of audio, the number of shots may refer to labelled events, each possibly varying in length. Depending on feature length (i.e. the portion of the signal covered by a single spectrogram), this may result in more than N spectrograms representing N events. For instance, if the 5 shots collectively span 5 seconds and each spectrogram represents 250 ms of audio, there will be at least 20 spectrograms representing the 5 events, comprising the support set.

FSL with audio has found application in acoustic scene classification [147, 172] and activity detection [131, 209]. In 2019, Pons et al. [147] found that models pretrained on

Audioset [60] and then fine-tuned exhibit strong performance in acoustic scene classification. They also observed that Prototypical Networks [177] trained from scratch, while performing slightly worse, still offer good few-shot learning results and outperform many baseline methods. The authors note that when data distributions do not match, using Prototypical Networks trained from scratch is a better choice than using a pretrained network.

Shi et al. [172] evaluated various meta-learning-based approaches, including Prototypical Networks [177], Meta-Opt [99] and MAML [57], alongside fine-tuning-based methods in a common acoustic classification task. Their results indicated that fine-tuning-based methods had the lowest performance, followed by MAML and Meta-Opt, with Prototypical Networks proving to be the top performer in their experiments.

In 2020, Wang et al. [209] applied different metric-based meta-learning methods, such as Siamese Networks [92], Matching Networks [203], Prototypical Networks [177] and Relation Networks [189], to sound event detection, specifically a keyword-spotting task. Keyword-spotting, which can involve using a unique keyword per user in conjunction with user voice characteristics, is well-suited for FSL, enhancing privacy without extensive adaptation or additional data. Notably, this work was a binary classification task, a 'one-vs-all' scenario. The authors discovered that simple data augmentation, like time-shifting (as opposed to time-warping), increased prediction robustness. In both binary and multiclass tasks, Prototypical Networks emerged as the top-performing approach. These findings in [147, 172, 209] begin to establish Prototypical Networks as the preferred FSL approach for audio applications.

In the field of bioacoustics, FSL has gained popularity, particularly due to challenges proposed by the DCASE community [125, 130]. A detailed description of the 2021 challenge can be found in Section 4.2. These challenges involve implementing few-shot systems for binary classification tasks. The specific task is to classify events (mammal or bird vocalisations) in an audio recording containing a single class of interest. The support set is constructed from the first five occurrences of the class of interest in each recording, with all subsequent audio forming the query set. The system's output comprises the onset and offset boundaries of events corresponding to the class of interest in that recording.

According to broader findings presented in [131], prototype-based meta-learning methods (Prototypical Networks and their variants) work effectively and are commonly employed among submitted systems. However, transductive inference-based methods exhibit impressive performance at the cost of increased complexity. The authors in [131] generally recommend that a system includes inference-time adaptation for optimal performance, but that a system without inference-time adaptation should be a widespread default.

2.4 Learnable Frontends

In speech and audio processing, researchers aim to extract information from audio signals for analysis. As the techniques used in audio signal processing have advanced, so too have the features extracted from audio signals. In the era of deep learning, systems capable of directly learning features from data have emerged. Typically, thanks to increased computational power and memory availability, and progress in image-processing tasks using deep learning, spectrograms are employed in audio tasks. These spectrograms can be adapted to a specific frequency scale like the mel scale or undergo compression, such as logarithmic compression, to reduce the dynamic range. However, using spectrograms necessitates careful consideration and tuning of hyperparameters, like window type, length and overlap. Alternatively, it may involve the use of fixed filterbanks and static compression methods, which may not be suited to the data. Learnable frontends aim to learn some or all of these aspects: filterbanks; windowing; compression; and any associated hyperparameters.

To the best of our knowledge, Jaitly & Hinton [77] were the first to learn representations from raw audio in 2011. They utilised Restricted Boltzmann Machines to model fragments of speech signals and create features, which would then be used as the input to a phone detection system. The authors acknowledged the difficulty in learning from raw waveforms, due to their high dimensionality. Although these learned representations did not surpass mel filterbank energies in performance, they did outperform other methods.

In 2012, Chu & Alwan [35] extend the Expectation-Maximisation algorithm to to estimate acoustic model parameters, including centre frequencies and filter bandwidths for cepstral coefficient calculation. Their work focused on bird species identification, highlighting the

potential inadequacy of the widely used mel scale for this task, prompting the exploration of alternative filterbanks.

From this point onward, deep learning becomes more prevalent. In 2013, Palaz et al. [137] recognised that the the *convolutional* aspect of CNNs are well-suited to processing temporal signals such as audio. The authors utilise CNNs to extract features from raw waveforms, with the first convolutional layers acting as filters on the waveform. Despite the unconstrained nature of these convolutions, which allowed for complex structures, they did not surpass the performance of the MFCC used in a baseline system.

In 2013 Andén & Mallat [1] introduced the deep scattering transform, a signal processing technique designed to extract translation-invariant features from data. This wavelet-based method consists of a series of layers, with deeper layers capturing higher-level, more translation-invariant features. When applied to audio, the first order scattering transform can compute a time-domain approximation of a filterbank spectrogram. Details of this approximation can be found in Section 5.2.2 and Equations 5.12– 5.17.

Following the introduction of the deep scattering transform and a method for time-domain approximation of filterbank spectrograms, frontends employing learnable filterbanks typically fall into two categories: frequency-domain-based and scattering transform-based [58]. Frequency-domain-based frontends apply their filters in the frequency domain, necessitating a prior transformation of the audio into a spectrogram. This transformation requires specific choices of hyperparameters, particularly concerning windowing and FFT length. The advantage of frequency-domain-based filterbanks is that certain filter shapes, such as rectangular or triangular filters, are more easily realisable in the frequency domain, as they have infinite impulse responses in the time domain.

On the other hand, scattering transform-based approaches operate entirely in the time-domain, and utilise the convolutional nature of CNNs, as noted by Palaz et al. [137]. Furthermore, as the first-order scattering transform approximates a filterbank spectrogram, the hyperparameters controlling the STFT can also be learned. Other time-domain-based approaches exist, which do not rely on the scattering transform. Instead, they adopt a convolution-based approach, focusing solely on filtering without windowing. These simplified

time-domain approaches still require design decisions related to windowing, framesize, and overlap. Both frequency-domain-based and time-domain-based methods have their advantages, but recently, time-domain methods, particularly those based on the first-order scattering transform, are becoming more prevalent.

Learnable frontends employing learnable filterbanks can be parameterised or learn filter coefficients directly. Filterbanks which are parameterised are constrained to a known filter shape, have less trainable parameters, and parameters have direct meaning, easing analysis. Filterbanks where filter coefficients are learned directly offer a large degree of flexibility and can improve performance, at the cost of increased complexity. More detailed discussion on the domain of operation (time-/frequency-domain) and on the usage of constrained/unconstrained filters can be found in Section 5.2.

An outline of the literature around frequency-domain-based (Section 2.4.1) and time-domain-based frontends (Section 2.4.2) is provided below. This is followed by an overview of some learnable compression methods in Section 2.4.3.

2.4.1 Frequency-Domain-Based Learnable Frontends

Frequency-domain-based learnable frontends involve the calculation of the STFT to transform a signal into the Time-Frequency domain, followed by frequency axis warping using filters implemented purely in the frequency domain. This is typically implemented via matrix multiplication using a linear transformation matrix, projecting the Fast Fourier Transform (FFT) bins in the STFT to some filterbank. In these frontends, the coefficients of the filterbank are learned, either directly or via parameterisation, to optimise the TF-representation of the data jointly with the classifier.

Early attempts at implementing frequency-domain-based frontends within deep learning were undertaken by Sainath et al. [160] in 2013. The authors initialise the filterbank to a triangular, mel filterbank with no constraints on filter shape, i.e. the filters are not parameterised. Results indicate that learning the filterbank yielded some improvements in an ASR task, with additional enhancements achieved through feature normalisation. Notably, learned filters in the low-frequency regions remained similar to the mel initialisation, while

mid-frequency filters exhibited multiple peaks, suggesting these frequencies are important to the model. High-frequency filters acted as high-pass filters instead of the wide-bandwidth bandpass filters they were initialised to. The authors also make the following keen observation, applicable to learnable frontends in general, "... this [mel] filterbank is not really an appropriate choice as it is not learned for the objective at hand" [160], summing up that although perceptual scales have been useful, they are not optimised for the data or task.

In 2017, Seki et al. [168] introduced learnable Gaussian filters in the frequency domain, parameterised by gain, bandwidth and centre frequency parameters for each filter. This approach added fewer parameters to the model and resulted in more interpretable filters. Comparisons were made between a fixed Gaussian filterbank, a triangular mel filterbank, and the optimised Gaussian filters. The triangular filters exhibited better performance than the fixed Gaussian filters, but the learned Gaussian filter outperformed both fixed filterbanks. However, no comparison was made with learned triangular filters. The study noted that most changes in filter properties were associated with the gain parameter, while the centre frequency showed minimal change from its initialisation. Information regarding changes in the bandwidth parameter was not provided, and the lack of deviation from initial centre frequencies is a common occurrence.

In 2020, Cheuk et al. [34] attempt to learn FFT kernels directly with nnAudio. This library was designed to allow GPU-based feature extraction, where operations could also be optimised via backpropagation, including FFT kernels and filterbanks. Both can be utilised, creating learned STFT representations and learned filterbank matrices. Strictly speaking, nnAudio encompasses elements of both time-domain and frequency-domain methods. While these learned kernels and filterbanks indeed outperformed their fixed counterparts in a pitch detection task, the authors note the increased computational demands incurred during kernel learning.

In 2021, López-Espejo et al. [111] found that learnable frequency-domain filterbanks did not yield a statistically significant performance enhancement in a keyword spotting task. Similar to the approach of Sainath et al. [160], these filters were not parameterised, and all coefficients were learned directly. The resulting filters exhibited minimal deviation from their

initial state, as observed in [168], with minor changes in filter shape and gain. Nonetheless, the authors highlighted that learned filterbanks might reduce redundant information and facilitate improved performance, particularly in situations with hardware constraints and a need for small-footprint models. Subsequent work in 2023 [110], revealed that using a reduced number of learned filters, as compared to a larger number of fixed filters in the filterbank, notably enhanced performance, especially in low SNR conditions.

FastAudio [58], introduced in 2021 by Fu et al., combines elements from Seki et al.'s work [168], by returning to parameterised filterbanks and revisiting the question of filter shape. The work again tests triangular vs. Gaussian filters, although it excludes the gain parameter, which was the parameter which changed most and presumably contributed most to the performance gains in [168]. Instead, FastAudio exclusively parameterises filters by centre frequency and bandwidth. This frontend is partly a response to scattering transform-based frontends such as LEAF (discussed below and in Section 5.2.4), which the authors contend are time consuming and computationally intensive. While it is true that scattering transform-based approaches consume more time, GPU hardware and CNNs are well suited to these operations. Furthermore, scattering-transform-based approaches eliminate the need for hyperparameter tuning regarding the STFT. FastAudio performs well on a speaker verification task, and verifies the finding in [168] that triangular filters slightly outperform Gaussian filters.

In 2022, Peng et al. [143] returned to directly learning filter coefficients. To address filter shape and initialisation concerns, the authors imposed a sparsity constraint on the resulting filterbank matrix. Two such constraints were applied: (1) each filter is expected to activate a few frequency components and (2) reduce others to zero while maintaining differentiation between filters. The filters were initially set to mel triangular filters, and the sparsity constraints led to filters that outperformed the fixed mel filters. Some variations in center frequency, bandwidth, and filter density in specific frequency ranges were observed, but the most significant change was in filter shape. These changes remained interpretable due to the sparsity constraints. Overall, this approach holds promise for frequency-domain-based frontends.

2.4.2 Time-Domain-Based Learnable Frontends

Time-domain-based approaches rely on convolution operations to create TF-representations from the input audio. The rise of CNN-based architectures has enabled efficient GPU-based convolutions. However, time-domain methods have certain limitations compared to their frequency-domain counterparts. Some filter shapes, like rectangular and triangular filters, cannot be feasibly implemented with a finite number of samples. Additionally, while convolution operations are less computationally intensive on a GPU, frequency-domain approaches often use matrix multiplication, which is more computationally efficient. Nevertheless, time-domain approaches, especially scattering transform-based methods, are gaining prominence, as they can also learn the necessary windowing function for creating a TF-representation to be fed into the network's backbone.

In 2015, Hoshen et al. [73] utilise the convolutional architecture to implement a bank of unconstrained filters, originally initialised to a gammatone filterbank. These filters had centre frequencies matching the mel scale, and using max pooling were used to learn features directly from the audio waveform. The work was initially designed for multi-channel audio, aiming to learn a set of bandpass beamformers with a perceptual, auditory-like frequency scale. Later, Sainath et al. [162] revised it for single-channel audio, with a greater focus on improving the audio's feature representation rather than relying on a perceptual scale like the mel scale. However, Sainath et al. observed that these learned features could match but not surpass the baseline Log-Mel spectrogram results.

In 2018, Ravanelli & Bengio introduced SincNet [151]. They noted that when learning directly from the waveform, the weights of the first convolutional layer were not intuitive, and though they made sense to the neural network, they did not align with human understanding of the signal or present an efficient representation. By constraining the first layer to the difference of two sinc functions, each filter became a near-rectangular bandpass filter, parameterised by the upper and lower cutoff frequencies. The only imposed constraints are that the lower cutoff frequency $f_1 \geq 0$, and that the higher cutoff frequency $f_2 \geq f_1$. Compared to unconstrained, non-parameterised CNN filters, the SincNet filters offered improved performance and faster convergence.

Zeghidour et al. [217] introduced Time-Domain Filter Banks (TD) in 2018, utilising the scattering transform [1] to create a TF-representation of the signal. In their initial experiments, the authors approximated a mel spectrogram before allowing the filterbank (initialised as a mel filterbank) and a lowpass filter to be learned. Notably, the filterbank in TD lacks parameterisation, with coefficients learned directly, offering increased flexibility at the expense of longer training times and reduced interpretability. Subsequent work on TD [218] further explores the lowpass filter in scattering transform-based approaches, showing that it improves performance over usage of max or average pooling. Analysis of the final learned filters in [217], reveals some changes from the mel initialisation, but the initial filterbank structure remains intact. A more detailed overview of TD is included in Section 5.2.2.

Inspired by SincNet [151], Noé et al. [129] propose the use of Gabor filters instead of windowed sinc functions. They justify this choice by noting that a Gabor filter provides the best compromise in time-frequency resolution. Gabor filters provide a Gaussian response in the frequency-domain, which as shown in [58, 168], do not perform as well as triangular filters, but they are more easily implementable in the time-domain. The Gabor filters show improved performance compared to the approximated rectangular filters of SincNet [151]. The authors use a complex Gabor filter for phase information, however this does n0t offer any advantage as the model only leverages the magnitude.

Zeghidour et al. introduced another scattering transform-based frontend in 2021 with LEAF [216], an "all-in-one, off-the-shelf" learnable frontend. LEAF extends TD by introducing a learnable PCEN layer for compression and normalisation (see Section 2.4.3). This is in contrast to other methods which make use of log compression or non-linear activation layers. LEAF also shifts from learning filter coefficients directly to a parameterised filterbank. LEAF utilises a Gabor filterbank characterised by centre frequencies and bandwidths, similar to [129]. The authors cite issues of instability, overfitting and lack of interpretability as reasons for adopting a Gabor filterbank. LEAF outperforms both TD and SincNet on a variety of speech, music and acoustic scene classification tasks, but is outperformed by TD on a BAD task, a result replicated in our work in Section 5.3.4. In 2022, Schlüter & Gutenbrunner [166] developed an optimised implementation of LEAF.

Similar to many other learnable frontends, LEAF is initialised to a mel filterbank. The author's note that the learned filterbank does not deviate substantially from this initialisation. Zeghidour et al. [218] and Sainath et al. [162] credit this as the mel scale being a strong initialisation, work in this thesis believes otherwise, that learnable frontends are sensitive to their initialisation (See Section 5.4). For more detailed information about LEAF, including the optimised implementation, refer to Section 5.2.4.

2.4.3 Learnable Compression

In contrast to learnable filterbanks, learnable compression remains a less explored area. Typically, TF-representations of audio involve either no dynamic range compression or static methods like logarithmic or n-th root compression. These static methods, particularly logarithmic compression, have drawbacks, as discussed in detail in Section 5.1.2. In summary, they tend to over-compress less informative parts of the signal, depend on signal loudness, and (in the case of logarithmic compression) have a singularity at 0.

Per-Channel Energy Normalisation (PCEN), introduced by Wang et al. [211] in 2017, addresses issues encountered in far-field keyword spotting. It serves as an alternative to static compression methods and offers the advantage of joint optimisation with the model. The authors found it to perform better on far-field sources compared to logarithmic compression. Empirical findings by Lostanlen et al. [115] also demonstrate that PCEN Gaussianises magnitude distributions and decorrelates frequency bands in the resulting TF-representation, which can aid performance. PCEN is a popular compression choice, and has been utilised in various tasks as well as being incorporated in other learnable frontends such as LEAF [216]. Section 5.2.3 provides an overview of PCEN's operation, while a detailed asymptotic analysis is provided in [115].

Liu et al. [108] attempt to learn channel-dependent power law-based compression, proposing to make the parameter governing the compression curve trainable. This adds one additional parameter per channel. Analysis of the compression parameters reveals that, on human speech, additional compression is applied to the low and high frequency regions. Interestingly, the final learned compression parameters remain similar, regardless of their initial values. The

channel-dependent strategy enhances performance in a speaker verification task, surpassing static power law-based and logarithmic compression methods. However, being power-law based, this approach is still susceptible to the problems mentioned above.

In 2022, Schlüter & Gutenbrunner [166] suggested replacing the PCEN component of LEAF with trainable parameter-controlled logarithmic compression, coupled with median filtering and batch normalisation along the temporal axis. This compression technique, labelled as 'L-M-TBN', is not proposed as a standalone learnable compression method or a complete replacement for PCEN but is part of a suggested LEAF modification aimed at enhancing throughput. The author's find that on some tasks, the performance is similar between L-M-TBN and PCEN, whereas for other tasks the performance suffers.

2.5 Datasets

This work employs several datasets. In this section, we present an introductory overview of each dataset, including its origin, purpose, and role within this work. Further details will be provided where relevant.

The majority of data of this work is bioacoustic data, bird audio in particular. As a result, four datasets featuring bird and animal vocalisations are included. Three of these bioacoustics datasets originate from various challenges, including DCASE [125, 187] and BirdCLEF [84]. The fourth dataset is NIPS4BPlus [124]. NIPS4BPlus and the BirdCLEF datasets are suitable for species identification tasks, while the DCASE datasets are ideal for activity detection.

Additionally, two other datasets are used in this work for tasks related to human speech. These datasets are TIMIT [59], which contains clean, read speech, and MS-SNSD [153], a dataset that includes both speech and noise. In this work we only utilise the noise portion of MS-SNSD as an additive noise source for TIMIT.

The following is a concise description of each dataset, with further details provided when relevant throughout the work.

- **DCASE2019 Challenge** – The DCASE2019 Bird Audio Detection challenge

dataset [187] includes data from three distinct sources, resulting in a total of 35,690 recordings. Each recording has a duration of 10 seconds and a sample rate of 44.1 kHz. Annotations at the clip level indicate whether bird activity is present or absent in a recording, a binary classification task. There is a class imbalance, with approximately 60% of recordings containing bird activity. This dataset is utilised in Section 5.3 to benchmark various learnable frontends on a BAD task.

- **DCASE2021 Challenge** – The DCASE2021 Few-Shot Bioacoustic Sound Event Detection challenge dataset [125] comprises of training, validation and evaluation sets, containing a variety of bioacoustic audio. The training set comprises four sets of data from distinct sources and is designed as a multiclass dataset. In contrast, the validation and evaluation sets employ binary annotations and stem from two and three different sources, respectively. All sets are labelled for the onset and offset times of events within their respective recordings. For comprehensive dataset details, refer to Section 4.2.2. This data is utilised in Chapters 4 and 6 to train few-shot learning systems.

- **NIPS4BPlus** – The NIPS4BPlus dataset [124] is a richly annotated bird audio dataset, incorporating detailed temporal annotations suited for species identification. It consists of 687 recordings sampled at 44.1 kHz, amounting to one hour of audio content. The dataset contains vocalisations from 51 different bird species, as well as recordings containing no bird activity. This dataset is utilised in Chapter 3 to train a low-resource bird activity detection system, with specific details on preprocessing and feature extraction found in Section 3.2.

- **BirdCLEF2021** – The BirdCLEF2021 training set [84] comprises high-quality recordings of varying lengths sourced from Xeno-Canto, an online repository of crowd-sourced bird recordings. Each recording has been annotated to identify the bird species present, making it well-suited for bird species identification. There are 397 species present in the dataset from across the Americas. There is a significant class imbalance, with 12 species having 500 recordings while 9 species have fewer than 25 recordings. This dataset is utilised in Section 5.4 to investigate the sensitivity of learnable frontends to their initialisation. More details are provided in Section 5.4.2.

- **TIMIT** – The TIMIT corpus [59] consists of clean, read speech from 630 speakers located in North America. This dataset has found applications in various aspects of speech research, and in this study, it is utilised for a voice activity detection task. The dataset undergoes additional preprocessing in this work, involving the introduction of variable-length periods of silence and noise, sourced from MS-SNSD. Section 5.4 combines this dataset with MS-SNSD to explore the sensitivity of learnable frontends to their initialisation. More details are provided in Section 5.4.2.
- **MS-SNSD** – The MS-SNSD noise dataset [153] consists of fourteen noise types, encompassing both stationary and non-stationary noise profiles. This dataset is coupled with TIMIT, where its primary purpose is to serve as an additive source of noise. As with TIMIT, it is utilised in Section 5.4 to investigate the sensitivity of learnable frontends to their initialisation, with more details provided in Section 5.4.2.

2.6 Relevance to this Work

This literature review serves an introduction to many key topics present in this work, including the production and perception of bird vocalisations, challenges in recording and monitoring, FSL and learnable frontends. The framework, development and methodology of these aspects has been outlined and some common themes, issues and gaps in research have been identified. This section discusses the relevance of these themes, issues and gaps to the research presented here.

In recent years, significant strides have been made in species agnostic BAD through the integration of deep learning, specifically leveraging CNN-based architectures [23, 67, 98, 104, 117, 216]. However, on-device inference and adaptation remains a desirable objective [184]. While large CNN-based models demonstrate impressive performance, their computational demands are high. Although optimisation efforts can enhance their efficiency [85], the challenge persists in enabling real-time operation on constrained hardware. When BAD is conducted on-site, algorithms must facilitate real-time execution on embedded devices, such as an ARU. CNNs may be unsuitable in a situation where computational and memory resources are limited, even with extensive optimisation.

To this end, this thesis explores lightweight classifiers, encompassing both deep learning-based and traditional machine learning-focused approaches. Chapter 3 examines leveraging specific attributes of bird vocalisations, such as the "performance limit" [145, 146] between amplitude modulation and frequency modulation. This leads to the development of the AMPS feature set, tailored for low-resource classifiers such as random forests. The proposed system and experiments in that chapter demonstrate that comparable or superior performance to a small CNN can be attained across most metrics using features and models with reduced computational costs, which are suitable for edge deployment.

FSL has emerged as a promising avenue for BAD and BSID. A survey of the literature identifies Prototypical Networks [177] as the most popular FSL method for audio applications [131, 147, 172, 209]. This popularity in part stems from their simplicity in training and deployment in the context of FSL. Most of the computational cost in Prototypical Networks is due to the embedding function, implemented as convolutional layers, while classification involves pairwise distance calculations from prototypes. This is an efficient operation, compared to neural network-based classifiers. A small embedding function could reasonably be implemented on constrained hardware; protypical networks have been shown to work well with shallower networks [31]. Furthermore in a FSL framework, it can be tuned for specific purposes with few labelled data.

Chapter 4 explores FSL in the context of the DCASE2021 Few-Shot Bioacoustic Sound Event Detection Challenge [125], offering an overview of the challenge, analysis of the submitted entry, and discussion of other high-ranking entries. Chapter 6 (Section 6.2) revisits the FSL topic, integrating it with learnable frontends, as discussed later. The result is the development of a few-shot learning system for bioacoustic activity detection, the first to integrate fully-trainable frontends within the model.

Several studies in BAD literature [7, 216] point out the fact that the mel scale, rooted in human perception, may not be optimal for bird audio tasks. This is echoed in broader literature on learnable frontends, suggesting that while perceptual scales serve a purpose, they may not always be ideal within a modelling framework [160]. In fact, there is no reason to believe that the mel scale is optimal [66], even on tasks involving human speech.

Learnable frontends provide an avenue for directly extracting features from the raw waveform or a spectrogram. Those featuring interpretable, parameterised filterbanks [58, 129, 151, 168, 216] are especially useful, as their learned parameters have direct meaning such as distribution of centre frequencies and filter bandwidths. However, it is acknowledged that less interpretable, unparameterised filters with more parameters can yield improved performance [216, 217]. In addition to learnable filterbanks, frontends incorporating learnable compression [107, 166, 211, 216] have also been shown to be useful in situations with far-field sources and noise contamination, prevalent in bioacoustics [114, 117]. Chapter 5, specifically Section 5.3, conducts the most up-to-date and comprehensive comparative analysis of traditional (non-learnable) and learnable frontends using identical datasets, tasks, and model architectures, focusing on a BAD task. Assessing the effectiveness of these frontends and their application to bird audio was a crucial step in the development of the previously mentioned few-shot learning system.

A recurring theme in learnable filterbank literature is the observed lack of movement from the filters' initialisation [24, 111, 166, 168, 216, 217]. This is contrary to what one might expect of learnable filterbanks that are optimised for a certain domain and task. It may be tempting to assume that if the learned filters closely resemble their initialisation (e.g. mel) then the initialisation was already well-suited for the task (e.g. speech recognition). Although this may hold true for certain tasks, it does not explain the same phenomena occurring with different filterbank initialisations, whether pyschoacoustic (such as the bark scale) or non-psychoacoustic (a linear initialisation). It also does not explain the lack of change for radically different audio signals, such as bird vocalisations. It is more likely that this is an issue with how these frontends are trained, or with the optimisation strategy. This phenomenon is explored, characterised, and quantified in Chapter 5, specifically Section 5.4, which presents new insights on the sensitivity of learnable filterbanks to their initialisation. Proposed mitigation strategies to the filterbank initialisation problem are discussed in Chapter 6 (Section 6.1).

Chapter 3

Low Resource Bird Activity Detection with AMPS

As discussed in Chapter 1, automatic and remote long-term monitoring of bird populations is of increasing importance for scientific research and conservation as ecological and environmental factors change in the coming years [83, 196]. As automated monitoring may take place over long periods of time and in remote areas, it is reasonable to assume that the processing power available to any remote recording station will be limited, and that connection to cloud based computing may not always be possible. As the usage of ARUs increase [188], the deployment of such devices operating continuously in remote areas of interest is an essential first step in species identification and population monitoring tasks [154]. This motivates the task of activity detection as a filtering step, marking areas of interest in audio that could then be sent for further analysis, with a specific focus on optimising these methods for embedded applications. If bird activity detection is to be performed on-site, any algorithms must be capable of running in real-time on an embedded device such as an ARU.

While deep learning-based approaches constitute the current state-of-the-art for this task (see Section 2.2.2), these models may be unsuitable for use with constrained hardware. Although such models perform well, their computational cost is high when compared with

the models and features proposed in this chapter. Instead, the work in this chapter utilises more lightweight models which are less computationally expensive, while aiming to limit loss in performance.

Specifically, this chapter introduces features based on amplitude modulation which have been previously deployed by the Institute of Acoustics (IOA) for the long-term monitoring of wind farms [76]. These features draw upon a correlation between a bird species' vocalisation frequency range and their trill rate, or amplitude modulation within a given call or sequence of calls [146]. This relationship motivates the use of these features. Additionally, the system makes use of spectral parameters [52, 53] and pitch based features. This feature set is denoted as AMPS (Amplitude Modulation, Pitch, and Spectral features).

This chapter is structured as follows: Firstly, Section 3.1 outlines the preprocessing and feature extraction pipeline, and provides details on their implementation. Subsequently, Section 3.2 provides details regarding the experimental setup, which includes an overview of the dataset, the machine learning methods employed, and the hyperparameters used in both feature extraction and classification stages. The results of these experiments are presented and discussed in Section 3.3, and are followed by some concluding remarks on this work in Section 3.4. The feature set, experiments and analysis in this chapter, were originally presented [4] in SiPS 2021.

3.1 Preprocessing and Feature Extraction

In this section, the preprocessing and feature extraction pipeline is described, alongside implementation details for each process. This section also contains a brief analysis on the range of frequencies in bird vocalisations, which informs certain design decisions. A visual representation of this pipeline can be seen in Figure 3.1, showing the signal path from input audio file, to extracted feature vector.

3.1.1 Identifying a Frequency Range for Bird Vocalisations

As a preliminary step in the development of this bird activity detection system, an investigation into the "typical range of bird vocalisations" was conducted. Specifically we are

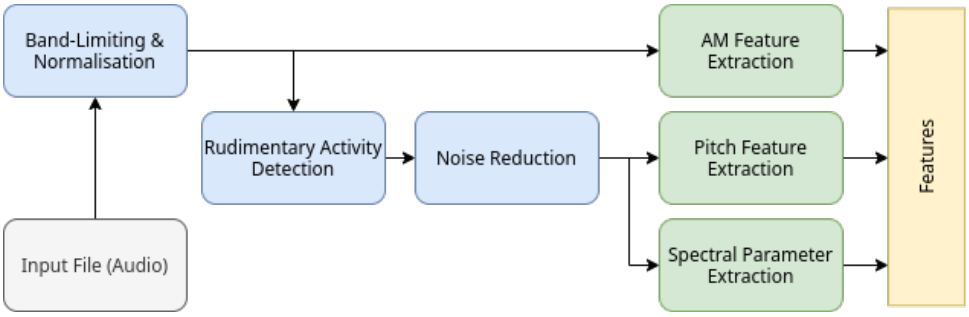

Figure 3.1: Flowchart of the feature extraction pipeline, with preprocessing steps. Common to all feature extraction algorithms is band limiting and normalisation, whereas pitch and spectral features are subject to additional activity detection and noise reduction. Activity detection and noise reduction applied before AM feature extraction will change the envelope of the signal. Blocks in blue indicate preprocessing operations, and green blocks indicate feature extraction operations.

interested in identifying a typical range of F_0 values, as the pitch and amplitude modulation features will be affected by the chosen range of values. This investigation also aimed to establish context for certain design choices in preprocessing and noise reduction, all of which require the definition of a relevant frequency band. The methods and findings which inform these decisions are briefly presented here. It is important to note that this analysis is indicative in nature, providing some understanding of the frequency ranges involved and justifying design decisions.

This analysis was carried out using YIN-bird, a modification of the YIN pitch tracking algorithm [41] proposed by O'Reilly et al. [134] in 2016. YIN-bird incorporates an adaptive frequency threshold for more accurate pitch tracking in bird audio. The parameters of YIN-bird were set in line with the original implementation outlined in [134].

Pitch analysis was conducted on the NIPS4BPlus dataset [124] utilised in these experiments and details of this dataset can be found in Section 3.2.1. Histograms were computed for each segment of audio containing bird activity, based on the estimated pitch samples generated by YIN-bird. The range of pitches defining the band of interest for subsequent preprocessing and feature extraction steps was informed by this histogram, shown in Figure 3.2.

This analysis revealed that many of the vocalisations in the dataset exhibit a fundamental frequency between 3.2 kHz and 4 kHz, with the mode centred on the bin spanning 3.6 kHz –

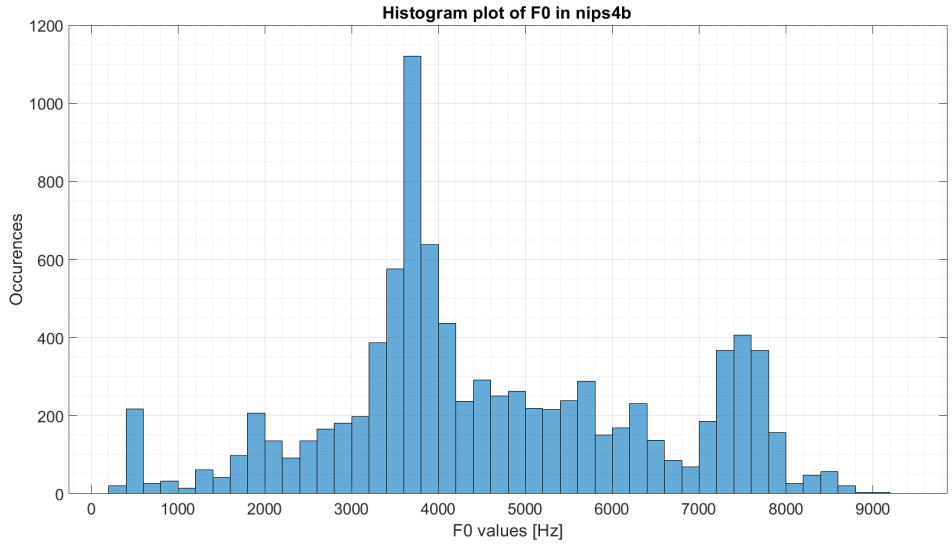

Figure 3.2: Histogram of F_0 values throughout the entire nips4b training set. Although there exist some occurrences below 800 Hz, we have decided to make the low frequency cutoff point 800 Hz to avoid contamination by human activity, or other environmental noise.

3.8 kHz. Consequently, the decision was made to define the broadband range of F_0 values for bird vocalisations as spanning 800 Hz and 9 kHz. This choice encompasses the majority of bird vocalisations while removing potentially unwanted sources of environmental noise. It also corroborates with other sources, which indicate fundamental frequency ranges from 1 kHz – 9 kHz for songbirds [158] and 100 Hz – 12 kHz for birds across all orders [63].

3.1.2 Preprocessing

Field recordings can be susceptible to contamination by various sources of noise including weather conditions, insects, and human activity. To enhance the quality of extracted features and improve classifier performance, some preprocessing of the audio is performed based on prior knowledge or assumptions about the data. These preprocessing steps encompass several processes, namely: band-limiting, normalisation, rudimentary activity detection, and noise reduction via filterbanks. As shown in Figure 3.1, the rudimentary activity detection and noise reduction steps are only performed prior to the extraction of pitch and spectral parameter-based features. Applying these processes before extracting AM features will contaminate the amplitude envelope of the signal, and corrupt the AM features.

Band limiting of the signal is implemented using a bandpass FIR filter, with the filter coefficients determined using the window method and a window size of 1024 samples. The cutoff frequencies of this bandpass filter are specified as 800 Hz for the high-pass frequency and 10 kHz for the low-pass frequency. This aligns with the decision outlined in Section 3.1.1 and allows for additional harmonic content to be included in the calculation of AM (see Section 3.1.3). Additionally, normalisation of the audio is incorporated to address the variation in audio levels within the recordings. As the recordings lack calibration for sound pressure level, the normalisation process involves rescaling the signal to span the entire dynamic range, or 0 dBFS. This normalisation is applied after band-limiting the signal.

Furthermore, an initial 'first-pass' activity detection is carried out on the signal. This method relies on calculating the short-time energy within a specified frequency band, which is then compared to the energy across the entire signal. The signal x is divided up into frames of length N. Each frame is a sequence $\mathbf{x}_{\text{slice}}$ defined in Equation 3.1. This frame is windowed with the windowing function w, and the energy of the frame $e_{\mathbf{x}_{\text{slice}}}$ is then calculated as in Equation 3.2. In these experiments: $N = 4410$, equivalent to 0.1 s, and w is a Hann window [69]. A Hann window with 50% overlap between frames is utilised in both activity detection and noise reduction processes to ensure unity gain.

$$\mathbf{x}_{\text{slice}} = \{x[n], ..., x[n+N]\} \tag{3.1}$$

$$e_{\mathbf{x}_{\text{slice}}} = \sum_{m=0}^{N-1} \left(x_{\text{slice}}[m]w[m]\right)^2 \tag{3.2}$$

The frequency band of interest can be isolated using the bandpass filter h, seen in Equation 3.3. If the ratio ϕ (Equation 3.5) between the energy in this specified band (Equation 3.4) and the energy contained in the rest of the signal (Equation 3.2) is above a certain threshold Φ (Equation 3.6), then that frame is marked as containing activity. The value of Φ is determined experimentally, alongside other preprocessing parameters. The resulting sequence of decisions undergoes median filtering with a window size of 5, incorporating a hysteresis effect, and the signal is reconstructed using frames marked as containing activity.

$$y[n] = \sum_{m=0}^{N-1} \mathbf{x}_{\text{slice}}[m]h[n-m] \tag{3.3}$$

$$e_{\mathbf{y}} = \sum_{m=0}^{N-1} \big(y[m]w[m]\big)^2 \tag{3.4}$$

$$\phi = \frac{e_{\mathbf{y}}}{e_{\mathbf{x}_{\text{slice}}}} \tag{3.5}$$

$$d_{\text{slice}} = \begin{cases} 1 & \phi > \Phi \\ 0 & \phi \leq \Phi \end{cases} \tag{3.6}$$

Noise reduction is achieved using a filterbank comprising 20 $1/6^{\text{th}}$-octave filters, with a reference frequency of 2 kHz, covering the range of F_0 values found in Section 3.1.1. The signal is once again partitioned into frames of length $N = 11025$, equivalent to 0.25 s, and windowed using a Hann window [69]. On a per-frame basis, the normalised filterbank energies of the signal are calculated. The signal is then reconstructed from the three filterbanks with the highest normalised energy. This technique, while crude, effectively eliminates most noise while preserving the integrity of bird vocalisations.

It is crucial that the activity detection and noise reduction processes are only applied prior to pitch and spectral feature extraction. Unlike band-limiting, which merely filters out information outside the specified frequency range of interest, the initial activity detection and noise reduction operations significantly modify the amplitude envelope of the signal. Applying these preprocessing steps prior to AM feature extraction would corrupt the AM features due to the altered amplitude envelope, as previously mentioned.

3.1.3 Feature Extraction

This system makes use or three categories of features: as illustrated in Figure 3.1: pitch features, spectral features and amplitude modulation features. Within the feature extraction process, four pitch features, four spectral features, and three AM features are computed.

The features and extraction methods employed are outlined in dedicated sections below. These features combined create an 11x1 feature vector for each 1-second frame of data. This feature vector serves as input to the classifier, used to determine whether the frame contains bird activity or not.

Pitch Features

The pitch features extracted are derived from pitch values collected over the 1-second analysis interval. The pitch extraction algorithm employed is YIN-Bird [134]. It is worth noting that prior to YIN-Bird, the unmodified YIN [41] pitch detection algorithm has been shown to be a more accurate pitch detection algorithm for birdsong than other popular methods [134]. The modification made by O'Reilly et al. to create YIN-Bird is the addition of adaptive parameterisation to identify a suitable minimum pitch value, or minimum frequency threshold. This adaptation method utilises information from the signal's spectrogram to dynamically adjust the minimum frequency threshold. This modification addresses the wider bandwidth and frequency modulation present in bird vocalisations, which differs from human speech.

During the 1-second analysis window, 50 samples of the fundamental frequency are extracted. This corresponds to a window length of 20 ms. The features obtained from this process are computed based on statistical moments pertaining to these pitch samples. The first four moments are computed: sample mean, sample variance, sample skewness and sample kurtosis. These moments collectively describe the shape and distribution of the sampled pitch values, offering an overall characterisation of pitch within the analysis window. It is important to note that by computing these moments, temporal information is discarded, as it is not utilised in the classification process.

In cases where there is no fundamental frequency, such as silence from the first pass activity detector or unvoiced sounds, those samples are represented as NaN. These NaN values are not included in the calculation of moments, instead, they are excluded from the analysis. This approach is adopted to prevent these values from artificially skewing the computed statistics, a situation that might occur if these samples are given a value of zero.

Spectral Parameters

Descriptive, low-level measures taken from the spectral domain are also used as input features. These descriptive measures aim to further characterise the sound within an analysis window, by providing insight into the energy distribution. Similar to the pitch feature extraction process, these features are computed within frames contained within the analysis window. The sample mean and sample variance values across the entire analysis window are then presented as features to the classifier. The same window length used in the pitch extraction process is utilised here, meaning that there are 50 samples within a given analysis window.

In these experiments, spectral centroid and spectral rolloff are employed. The spectral centroid represents the weighted mean of frequencies present in a frame, effectively depicting the 'center of mass' of the spectral frame. Given that the STFT of the entire signal is calculated, this corresponds to a weighted mean of each column, yielding a vector of spectral centroids for the entire signal. From this vector, the sample mean and variance for each overlapping analysis window can be calculated. Spectral rolloff is the frequency below which a specified percentage of energy is contained (e.g. 95%). It provides insights into the signal's bandwidth and can serve as an indicator of harmonic content when it deviates from the fundamental frequency. Per 1-second analysis window, the sample mean and sample variance of spectral rolloff is also calculated. The algorithms used to calculate both spectral centroid and spectral rolloff are detailed in [171].

Amplitude Modulation Features

The method for calculating AM and deriving features from it takes inspiration from the work IOA AM Working Group, whose research focused around detecting amplitude modulation caused by wind turbines [76]. This algorithm, which detected and characterised AM, has been adapted to suit this application. Notable changes to the calculation of AM in this use case include the use of shorter window lengths (10 ms instead of 100 ms to account for the higher rate of AM) and the frequency bands for which AM is calculated. For this application, AM based features are computed across four two-octave bands: 500 Hz-2 kHz, 1 kHz-4 kHz,

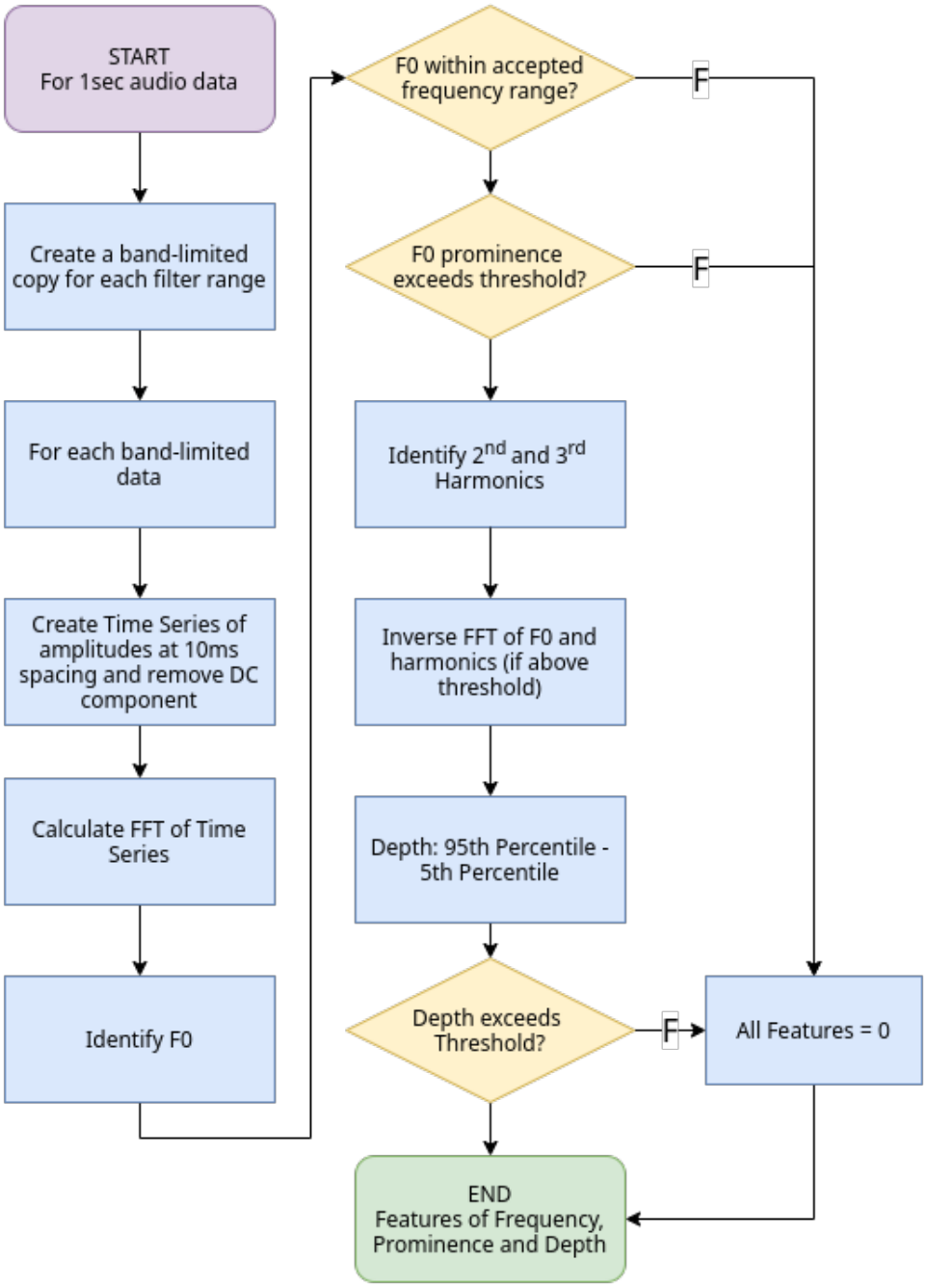

Figure 3.3: Flowchart of Amplitude Modulation Feature Extraction, adapted from the IOA method. Whether AM is detected or not is dependent on meeting criteria for 'valid AM frequency', 'AM prominence' and 'AM depth'. These comprise the extracted AM features used in the classifier.

2 kHz-8 kHz, and 4 kHz-16 kHz (band pass filtering has removed most information above 10 kHz). This allows for total coverage of frequencies which the signal had previously been band-limited to in Section 3.1.2. The band which exhibits the most AM, is the band where the features are taken from.

The algorithm for extracting AM-based features is presented in the flow chart in Figure 3.3. These features are AM Frequency, the prominence of the fundamental frequency of AM in relation to the surrounding frequency bins, and the modulation depth of AM as calculated from a reconstructed signal. Information indicating the presence or absence of AM in a given window is provided by non-zero values within these features. If all features are 0, then no AM was found in the signal. Consequently, this algorithm yields three features.

Let each 1-second window of data be denoted by x. First, the sequence x_i is calculated for each of the two-octave span frequency bands. The sequence x_i results from filtering the signal x with the bandpass filter h_i (Equation 3.7). For each x_i, the sequence is squared and the expected value taken over a period of M samples. In these experiments, $M = 441$. This results in a downsampled sequence of energy values, the square root of which is the AM envelope sequence for each band $\mathbf{x^i_{env}}$ as in Equation 3.8. This slice of data corresponds to the band-limited, 1-second window on which AM features are calculated.

An example of bird audio, and the extracted envelope of the signal in the 500 Hz-2 kHz band, can be seen in Figure 3.4. This figure also includes spectrograms of the two signals. Note the high energy at 0 Hz in the spectrogram of the extracted envelope. The 0 Hz component of this time series $\mathbf{x^i_{env}}$ is removed via subtracting the mean value of the signal, as the 0 Hz frequency component is not included in the feature extraction process.

$$x_i[n] = \sum_k x[n-k]h_i[k] \tag{3.7}$$

$$x^i_{env}[n] = \sqrt{\frac{1}{M}\sum_{m=0}^{M-1} x_i^2[nM+m]} \tag{3.8}$$

Subsequently, this envelope time series $\mathbf{x^i_{env}}$ is transformed into a power spectrum P^i_{env}, as

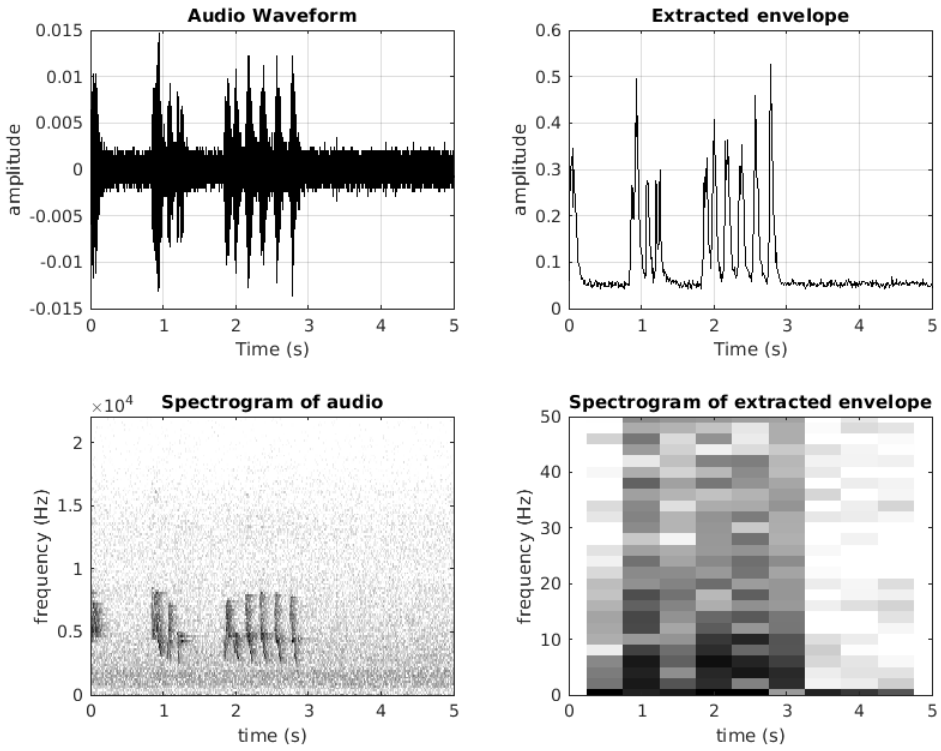

Figure 3.4: Example of an audio file in the NIPS4BPlus dataset containing bird vocalisations and the extracted envelope prior to removal of the DC component. Spectrograms of the audio and envelope are also shown.

described in Equation 3.9, and peak detection is performed to identify the index of the frequency bin containing the most energy, K_p^i. This index corresponds to the fundamental frequency of the amplitude envelope (Equation 3.10).

$$\begin{aligned} X_{env}^i &= \mathcal{F}(x_{env}^i) \\ P_{env}^i[k] &= \frac{1}{N^2}|X_{env}^i[k]|^2 \qquad (3.9) \\ K_p^i &= \underset{k}{\operatorname{argmax}}\left(P_{env}^i[k]\right) \qquad (3.10) \end{aligned}$$

To mitigate the possibility of misidentifying AM activity, a peak prominence criterion is employed. This criterion calculates the ratio of the peak's amplitude to the expected value of the three surrounding bins, as detailed in Equation 3.11. If this calculated peak prominence

exceeds a certain threshold, the frequency associated with that index is considered the fundamental frequency of amplitude modulation.

$$Q^i_{peak} = \frac{P^i_{env}[K^i_p]}{\frac{1}{|N|}\sum_{n\in N} P^i_{env}[K^i_p + n]} \tag{3.11}$$

$$\text{where, } N = \{\pm 1, \pm 2, \pm 3\}$$

Usage of this criterion helps to identify pronounced peaks in the frequency domain, as indistinct peaks could be classified as noise. If this peak is particularly prominent, the second and third harmonics are also checked using the same method. Should these harmonics also be deemed prominent by the process above, they are included in the reconstruction of the envelope employed to calculate the depth of modulation. The set of the frequency indices used in reconstruction of the signal is denoted as $\mathbf{K}^i_{peaks}$.

All valid peaks, which encompass the fundamental frequency and potentially the second and third harmonics, are kept whilst all the other frequency bins are set to a value of zero. This new spectrum is transformed back into the time domain as the signal y^i as shown in Equation 3.12. While this can introduce ringing artefacts to the reconstructed signal, potentially altering the modulation depth from that of the original signal, it is important to note that since this process is applied to all AM features, it is not expected to have any impact on classification. It is from this reconstructed signal that the amplitude modulation depth D^i is calculated:

$$\mathbf{y^i} = \mathbb{R}\{\mathcal{F}^{-1}(X^i_{env}[\mathbf{K}^i_{peaks}])\} \tag{3.12}$$

$$D^i = P_{95}(\mathbf{y^i}) - P_5(\mathbf{y^i}) \tag{3.13}$$

This value is calculated from the 95$^{\text{th}}$ and 5$^{\text{th}}$ percentile values of the reconstructed signal, as per the IOA method. Thus D^i represents the modulation depth of AM for band i. If AM is not detected in an analysis window, a value of zero is assigned to AM Frequency, AM

Prominence and AM Depth. Once again, the frequency band which exhibits the most AM, is the band where the features are drawn from.

3.2 Experimental Setup

The following experiments involve a Bird Activity Detection (BAD) task. As previously mentioned, BAD plays a crucial role as an initial filtering step for any potential off-site analysis of bird audio recordings (Section 2.2.2). In this specific context, the task of activity detection involves binary classification, aiming to determine whether a 1-second segment of audio contains bird vocalisations or not. It should be noted that BAD is a species agnostic task, meaning that the system is expected to generalise its performance to various species and types of vocalisations that may not have been encountered in the training dataset.

This section is divided into two subsections. Firstly, in Section 3.2.1, a description of the dataset used throughout these experiments is provided, namely NIPS4BPlus [124]. Secondly, Section 3.2.2 provides an outline of the classifiers employed in these experiments, details of feature extraction and classifier hyperparameters, and introduces a compact CNN-based system for comparative analysis.

3.2.1 Data

The dataset employed in this study is the NIPS4BPlus dataset, a richly annotated birdsong audio dataset released by Morfi et al. [124] in 2019. This dataset combines previously released training data in the form of bird vocalisations, from bird classification challenges [61], and incorporates detailed temporal annotations centred around the species identification task. These finely grained, temporal annotations make this dataset highly suitable for this work, where the aim is to detect activity in windows of one second. Many alternative datasets lack these fine-grained temporal annotations and instead provide activity labels at the clip level.

This dataset comprises 687 recordings sampled at 44.1 kHz, encompassing one hour of audio content featuring vocalisations from 51 different bird species. The dataset also contains recordings in which no birds are present (100 files), as well as human and animal activity

throughout. Vocalisations span a broad range of fundamental frequencies, with the majority of energy contained between 1 kHz and 8 kHz (see Section 3.1.1). These recordings took place across 39 locations in France and Spain, starting 30 minutes after sunrise and continuing for an additional 3 hours.

As part of this work, the labels have been converted to indicate bird activity in 1-second windows with 50% overlap. The result of this segmentation is 9 windows per file, covering each 5-second audio clip. The labels represent a binary classification, discerning whether there is bird activity within that window, regardless of vocalisation type or bird species. The resulting labels resemble those in datasets such as freefield1010 and warblrb10k, both released by Dan Stowell as part of the DCASE Bird Audio Detection Challenges [187]. However, in these datasets, labels are provided at the clip level, with a single label assigned per 10-second audio clip. Bird activity is present in approximately 40% of the 1-second windows. The results provided in this chapter include a breakdown of selected metrics for both the 'bird present' and 'bird absent' classes.

3.2.2 Feature Extraction and Classification

Feature extraction follows the procedures detailed in Section 3.1.3, yielding an 11x1 feature vector for each analysis window. As previously mentioned, this feature set is denoted as AMPS, representing Amplitude Modulation, Pitch, and Spectral features. The parameters governing feature extraction can be found in Table 3.1 within the relevant section of the table. Details are provided under the relevant headings in Section 3.1.3. AM features will have a fundamental frequency between 1-10 Hz, the peak of the fundamental frequency is required to be three times that of the surrounding bins, and the feature must have a minimum modulation depth of 0.01. Pitch features and spectral features are computed using the same window length and overlap, utilising Hann windows. The parameters influencing feature extraction were determined through factorial experiments following hyperparameter tuning of the classification models.

Classifier selection was driven by the suitability of each algorithm for embedded applications, with an emphasis on low memory and computational resource requirements. Consequently,

logistic classification, SVM, and Random Forest classifiers were evaluated. During model training, the primary objective was to maximise the F1-Score. However, when evaluating the performance of the classifiers, other metrics such as accuracy are also reported.

Logistic classification serves as a baseline classifier, with the classifier's threshold adjusted to optimise the F1-score. SVMs have been used previously by Fagerlund [52], and our model has been configured similarly. Random forest classifiers, known for their effectiveness in birdsong analysis as demonstrated by Stowell and Plumbley [186], were configured with 500 trees. However, it is worth noting that performance is not significantly reduced with smaller forests, allowing for potential computational savings.

Additionally, a stacking classifier [212] is also implemented, an ensemble technique that combines the outputs of the logistic, SVM, and Random Forest classifiers. Predictions from each individual classifier are stacked together and serve as input to a final meta-classifier. This meta-classifier, another instance of a SVM classifier, is trained using cross-validated outputs from the stacked classifiers. The stacking method aims to reduce the bias of any one classifier.

In contrast to these more traditional machine learning classifiers, a basic CNN-based system specially tailored for this task is also evaluated. The architecture and training of this network draw inspiration from the work of Grill and Schlüter [67]. This CNN utilises mel-frequency spectrograms as the input feature. The construction of the mel-frequency spectrogram is guided by the same parameters as specified in [67]. This results in a Time-Frequency representation composed of 80 frames and 80 mel filters, effectively rendering the spectrogram as a feature matrix of dimensions $\mathbb{R}^{80\times80}$.

To assess classifier performance, an 80/20 split of the dataset is performed, creating training and test sets. Classifier hyperparameters and decision thresholds undergo tuning via grid search techniques five-fold cross-validation on the training set. Threshold adjustments for the Logistic Classifier were made in increments of 0.01 and through tuning, was decreased to 0.45. Decision thresholds for the SVM and Random Forest classifier were not tuned and kept fixed at 0.5. It is acknowledged that fine-tuning the thresholds of these classifiers could have potentially enhanced the performance of both methods. Class balance is maintained with

Classifier	Hyperparameter	Value
Logistic Classifier	Threshold	0.45
	Solver	LBFGS
SVM	Kernel	RBF
	Regularisation Param.	1.0
	Gamma	0.5
Random Forest	Criterion	KL Divergence
	Max. Depth	8
	Min. Samples	8
	Features per Node	4
	N. Trees	500
Feature	**Hyperparameter**	**Value**
AM Features	Min. Modulation Freq. [Hz]	1
	Max. Modulation Freq. [Hz]	10
	Prominence Cutoff	3
	Depth Threshold	0.01
Pitch Features	Window Length [s]	0.02
	Window Overlap [s]	0.01
	Threshold	0.3
Spect. Features	Window Length [s]	0.02
	Window Overlap [s]	0.01

Table 3.1: Final hyperparameter values for Logistic, SVM and Random Forest classifiers. These were determined using a grid search over the relevant parameters. Below are the parameters used in the feature extraction of amplitude modulation, pitch and spectral features.

Classifier	**Acc.**	**F1-Score**	**Prec.**	**Recall**
Logistic Classifier ($\theta = 0.45$)	0.538	$\mathbf{0.611}$	0.462	$\mathbf{0.900}$
SVM	0.704	0.574	0.685	0.493
Random Forest	$\mathbf{0.721}$	0.604	$\mathbf{0.706}$	0.527
Stacking Classifier	0.713	0.594	0.693	0.520

Table 3.2: Bird Audio Detection with AMPS features, evaluated using a Logistic Classifier, Support Vector Machines, Random Forests and a Stacking Classifier. Best results for each metric are marked in **bold**.

approximately 40% positive instances and 60% negative instances (see Section 3.2.1), both in the dataset split and within each fold used for cross-validation. In Table 3.1, a detailed breakdown of the hyperparameters for each classifier is provided. These hyperparameters are optimised to maximise the F1-Score of models, although other metrics are taken into consideration later in Section 3.3. Reported results are based on the performance of each model on the test set.

3.3 Results and Discussion

Table 3.2 reports the performance of the Random Forest, SVM and the baseline Logistic Classifier. In this work, F1-Score is prioritised, as both precision and recall are valued. To this end, values for precision, recall and F1-Score are reported, in addition to model accuracy.

The Logistic Classifier achieves the highest F1-Score (0.611) when using a threshold parameter of $\theta = 0.45$. However, it exhibits poor accuracy (0.538) and precision (0.462) rendering it unsuitable for this binary classification task. The Logistic Classifier demonstrates a strong bias toward the positive 'bird present' class, leading to numerous false positives in its classification output. The SVM exhibits a lower F1-Score (0.574), but performs significantly better in terms of accuracy (0.704) and precision (0.574). However, it suffers from poor recall (0.493).

The Random Forest emerges as the top-performing classifier overall, with the highest accuracy (0.721), precision (0.706) and only marginal degradation in F1-Score when compared with the Logistic Classifier (0.604 for the Random Forest vs. 0.611 for the Logistic

Class	F1-Score	Precision	Recall
Bird Absent	0.785	0.727	0.852
Bird Present	0.604	0.706	0.527
Weighted Average	0.712	0.719	0.721

Table 3.3: Random forest classification using AMPS features, broken down by the two classes in the activity detection task: Bird Absent and Bird Present.

Feature Set	Accuracy	F1-Score	Precision	Recall
MFCCs	0.691	0.561	0.655	0.491
CNN	0.674	**0.643**	0.677	**0.612**
AMPS	**0.721**	0.604	**0.706**	0.527

Table 3.4: Comparison of the AMPS feature set to MFCCs when using Random Forests, and to the pruned and quantised CNN model using mel-spectrograms. Best results for each metric are marked in **bold**.

Classifier). Nevertheless, the Random Forest also suffers from poor recall (0.527). The stacking classifier delivers respectable performance relative to the individual classifiers but does not surpass the Random Forest.

Of the evaluated classifiers, Random Forests yield the best overall performance with the AMPS feature set. Furthermore, Table 3.3 shows class-specific metrics for the Random Forest with AMPS features. Given the dataset's slight imbalance towards the negative class, these results allow for a more thorough analysis of the classifier's performance for each class. The higher scores for the 'Bird Absent' class align with expectations due to the larger number of training examples for this class. However, the scores for the 'Bird Present' class and the overall metric averages, show the Random Forest classifier's capability to distinguish between the two classes.

In instances the classifier incorrectly labelled a window as 'Bird Absent', the audio typically contained 'unvoiced' calls, characterised by little harmonic content or had significant amounts of noise, attributed to factors such as wind, insects, or human activity. This issue of noise affecting features is a recurring theme in bird audio and bioacoustics more generally, and is discussed in more detail in Chapter 5.

At the time this work was carried out, prevailing state-of-the-art methods involved the use of

Classifier	**No. of Ops**
Random Forest	16k CMPs
CNN	187k FLOPs

Table 3.5: Number of operations for pruned and quantised CNN versus Random Forest system.

Mel-Frequency Cepstral Coefficients (MFCC) features or employing CNNs with spectrograms as input. Consequently, Table 3.4 presents a comparison of the Random Forest classifier's performance when using the AMPS feature set versus MFCCs as input. Specifically, the first 13 MFCCs were used, with the algorithm outlined in [171]. Additional information regarding MFCCs can also be found in Section 5.1.1. The AMPS features outperform MFCCs on this task, across all metrics. MFCCs are susceptible to corruption from additive noise, which is prevalent in remote field recordings.

The CNN has a higher F1-Score, but the performance is not dramatically higher, as might initially be anticipated. A closer examination of the CNN's predictions reveals that the CNN performs better on the 'unvoiced' vocalisations and signals containing in-band noise. In contrast, the AMPS features, when coupled with Random Forest classification, demonstrate better performance on fainter, far-field signals which are uncorrupted by noise. Both systems encounter difficulties on recordings with very poor SNR.

The computational cost of each model is considered in Table 3.5. Measurement of CNN complexity in terms of floating point operations required is in line with the work of Tan et al. in [190]. Additionally, for edge computing applications, the model undergoes pruning and quantisation, achieved through TensorFlow's built-in pruning, quantisation and optimisation functions. Random forests' complexity is measured by the number of comparisons conducted during classification, employing 500 trees, a maximum tree depth of eight, and four features per node. This results in a worst-case scenario of 16k comparisons, in contrast to 187k FLOPs for the pruned CNN-based approach.

The difference in efficiency between these two operations (CMP vs various other floating-point operations) is contingent on the processor architecture. However, a MAC Operation will be greater than, or equal to, a CMP operation in terms of machine cycles.

Therefore, the model is at least 10 times less expensive computationally. Thus, overall this approach offers a huge computational saving, with a limited sacrifice in performance.

3.4 Moving towards Few-Shot Learning

This chapter introduced an approach to species-agnostic bird activity detection, using low-resource classifiers, in conjunction with the AMPS feature set. It also outlines a method of calculating AM-based features, to complement the features drawn from pitch data and spectral representations. The algorithms used are well within the capabilities of embedded devices, delivering performance that is only slightly below that of a small CNN but at a fraction of the computational expense. This approach demonstrates promise as an initial filtering step, aiding ornithologists in remotely monitoring bird populations, and reduces the amount of data to be stored and processed off-site.

This research was conducted at the start of this PhD. At the time, low-resource activity detection capable of running directly on ARUs held significant potential. It continues to be an attractive area of study, as developing systems capable of autonomous operation offers many advantages. However, as this work was concluding, few-shot learning for bioacoustics began to emerge as a cutting-edge technology.

Few-shot learning is particularly appealing for bioacoustics due to the limited availability of large, fully annotated datasets and its capacity for test-time adaptation and generalisation. Some, though not all, few-shot learning frameworks also offer the potential for computational efficiency, providing an additional advantage. This approach is consistent with the guiding principle in this thesis of developing low-resource systems, in this case applicable to both data and processing power. Moreover, few-shot learning enables the use of neural architectures, with the hope of achieving higher performance.

As a result, the focus of this PhD research shifted toward the exploration of few-shot learning, which is detailed in Chapter 4. This work on few-shot learning further underscored the significance of feature representation and the impact of noise and dynamic range on input features. These insights serve as the foundation for the subsequent chapters on learnable frontends, discussed in Chapters 5 and 6.

Chapter 4

Bioacoustic Event Detection with Prototypical Networks

The emergence of powerful computing devices, large labelled datasets and advanced architectures have made deep learning highly successful in various tasks. This is also true in the realm of bioacoustics [127, 184], where the flexibility of deep learning can be applied to many tasks including activity detection and species identification. However, traditional supervised learning techniques rely on large amounts of labelled data to attain high performance, and many techniques do not generalise from few examples. FSL is a paradigm which aims to learn from a limited number of supervised examples [207].

This thesis chapter is organised as follows: First, Prototypical Networks [177] are explained in Section 4.1 and details provided on their training and operation in FSL scenarios. This will be followed by an overview of the DCASE2021 Few-Shot Bioacoustic Sound Event Detection Challenge [125] in Section 4.2, which was entered into by the author of this thesis using a protonet-based entry. This section also includes details of the submitted system, which ranked 5th overall. The top performing systems are also discussed briefly with reference to the submitted system. The results presented until this point were published in [2]. Further analysis of the submitted system and attempts to improve performance are discussed in Section 4.3.

4.1 Few-Shot Learning and Prototypical Networks

Section 2.3 provides an overview of FSL including some approaches to classification in a few-shot setting. These approaches aim to generalise to new classes not present in the original training set, given only a few examples of each new class. The challenge faced by these approaches is the risk of overfitting to the new data given so few examples, hindering their ability to generalise to other samples of the same class during inference.

Among the various proposed methods, Protoypical Networks (or protonets) were introduced by Snell et al. [177] in 2017 to address this fundamental issue of overfitting. These networks incorporate a simple inductive bias, similar to that found in k-nearest neighbours classification. This bias is based upon the concept of learning an embedding space, where points belonging to a particular class cluster around a prototype representation of that class.

As discussed in Section 2.3.2, protonets have found widespread usage in audio applications [147, 172, 209], outperforming other FSL methods. Protonets stand apart from other methods due to their straightforward, intuitive mechanisms and ease of training, which sets them apart from transductive methods which require additional test-time adaptation. Furthermore, protonets also have no restrictions on model size or architecture, which allows for low-complexity architectures to be employed. Additionally, protonets employ a meta-learning approach called episodic training [203]. This technique simulates the test time scenario by exposing the network to a limited number of examples, promoting efficient learning from few data while also mitigating the risk of overfitting by presenting the model with a diverse range of tasks.

The combination of the straightforward inductive bias, intuitive training strategy, efficiency, and the absence of restrictions on model architecture have contributed to the appeal of protonets as a promising approach for FSL with audio. These qualities have motivated the use of protonets as the method of choice for FSL tasks in this thesis.

In this section, a detailed explanation of how protonets operate is provided in Section 4.1.1. Furthermore, the training process, which encompasses episodic training and the loss function, is discussed in Sections 4.1.2 and 4.1.3.

4.1.1 Protonets

Prototypical Networks, as opposed to being a specific model architecture, serve as a framework designed specifically for FSL tasks. In a few-shot classification scenario, a small *support set* (denoted as S) consisting of N labelled examples per class is provided, and the objective is to classify the elements of the *query set* Q. To explain further, let S_k denote the subset of examples belonging to class k. Protonets utilise a non-linear mapping via an embedding function f_θ, where the feature vector $\mathbf{x} \in \mathbb{R}^D$ is mapped to a M-dimensional vector, i.e. $f_\theta : \mathbb{R}^D \to \mathbb{R}^M$, where θ are the learnable parameters of the network. Typically, the model architecture of f_θ is a convolutional encoder.

$$\mathbf{c_k} = \frac{1}{|S_k|} \sum_{\mathbf{X_i} \in S_k} f_\theta(\mathbf{X}_i) \tag{4.1}$$

$$p(y = k|\mathbf{X}) = \frac{\exp(-d(f_\theta(\mathbf{X}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\theta(\mathbf{X}), \mathbf{c}_{k'}))} \tag{4.2}$$

The prototype representation $\mathbf{c}_k$ of a class k is constructed from its *support set* S_k and represents the mean vector of the support embeddings, as defined in Equation 4.1. To determine the probability distribution of a query point $\mathbf{X}$ belonging to class k, the distances from the embedded query point to each class prototype are computed. This computation involves utilising some distance function $d(\cdot, \cdot)$ and applying the softmax operation over the distances, as described in Equation 4.2. An illustration of the construction of prototypes from elements of the support set and classification of query points, as shown by the decision boundaries and shaded regions in the figure, can be seen in Figure 4.1. This figure presents a scenario involving three classes, where the learned embedding function effectively clusters data points of the same class together while maintaining separation from other classes.

Although any distance function may be chosen for $d(\cdot, \cdot)$, there are strong arguments for choosing Bregman divergences [18, 28]. Let $\Phi : \Omega \to \mathbb{R}$ be a continuously differentiable, strictly convex function defined on the convex set Ω, the Bregman divergence for the function Φ for $x, y \in \Omega$ is defined by Equation 4.3. Bregman divergences are not true metrics, although they satisfy many properties of metrics such as non-negativity and a

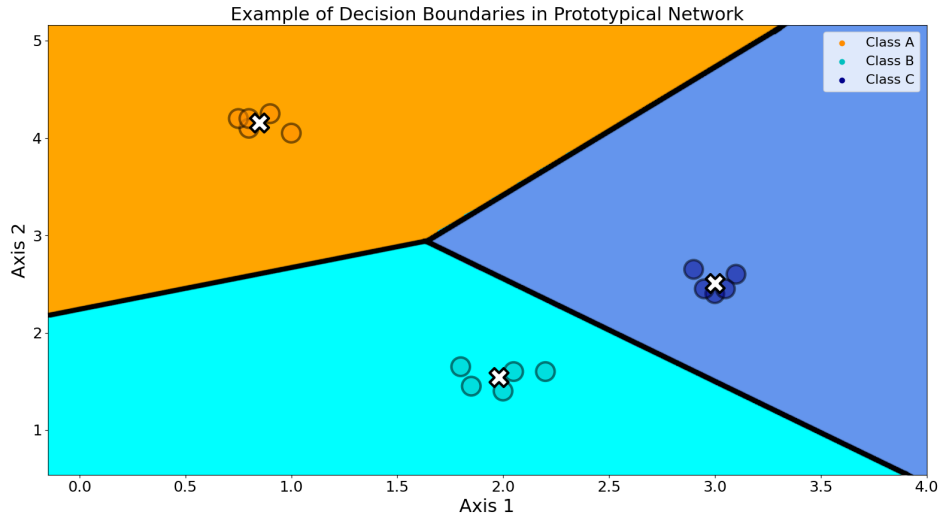

Figure 4.1: Prototypical Networks in a FSL scenario with a three class problem. It is important to note that the axes, labelled as 'Axis 1' and 'Axis 2', are arbitrary and do not hold specific or meaningful units. This simplified representation is used for illustrative purposes only. Prototypes for each class, denoted by a white 'X', are computed as the mean of embedded support points for each class. As query points are classified based on distance to class prototypes, the decision boundaries of each class are also shown.

unique zero (i.e. the divergence is only equal to zero when $x = y$) [9]. The mean or class prototype of a set of points, specifically the support points of a class, is the point that minimises the sum of the Bregman divergences between itself and the other points in the set [8]. Essentially, this mean point minimises the sum of dissimilarities from the other points based on the chosen convex function Φ.

$$D_\Phi(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) - \Phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\Phi(\mathbf{y}) \rangle \tag{4.3}$$

Consequently, the mean point takes on the role of the prototypical point, serving as the reference point for classifying query instances as belonging to that class. Several Bregman divergences exist, among them the Squared Error (L2 loss) and the Kullback-Leibler Divergence. The original implementation of protonets utilises the squared Euclidean distance, the simplest and most widely used of the Bregman divergences.

The squared Euclidean distance can be shown to be a Bregman divergence by the following proof. First, it must be acknowledged that the squared Euclidean norm corresponds to the dot product of a vector $\mathbf{x}$ with itself. Subsequently, the dot product of $\mathbf{x}$ with itself is equivalent to the inner product of $\mathbf{x}$ with itself, as shown in Equation 4.4.

$$\|\mathbf{x}\|^2 = \mathbf{x} \cdot \mathbf{x} = \langle \mathbf{x}, \mathbf{x} \rangle \tag{4.4}$$

Because the squared Euclidean norm is both differentiable and strictly convex, it is a suitable candidate for the function Φ. From the definition of a Bregman divergence, we require three terms: $\Phi(\mathbf{x})$, $\Phi(\mathbf{y})$ and $\nabla\Phi(\mathbf{y})$. Expressions for $\Phi(\mathbf{x})$ and $\Phi(\mathbf{y})$ are provided in Equations 4.5 and 4.6, these are both the inner product of the relevant vector with itself. Additionally, the gradient of $\Phi(\mathbf{y})$ is determined to be $2\mathbf{y}$, as shown in Equation 4.7.

$$\text{Let, } \Phi(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x} \rangle \tag{4.5}$$

$$\Phi(\mathbf{y}) = \langle \mathbf{y}, \mathbf{y} \rangle \tag{4.6}$$

$$\begin{aligned} \nabla\Phi(\mathbf{y}) &= \frac{d}{d\mathbf{y}} \langle \mathbf{y}, \mathbf{y} \rangle \\ &= \langle \mathbf{1}, \mathbf{y} \rangle + \langle \mathbf{y}, \mathbf{1} \rangle \\ &= \langle \mathbf{1}, \mathbf{y} \rangle + \langle \mathbf{1}, \mathbf{y} \rangle \text{ (Symmetry)} \\ &= \langle \mathbf{2}, \mathbf{y} \rangle \text{ (Linearity in first argument)} \\ &= 2\mathbf{y} \end{aligned} \tag{4.7}$$

Substituting the three terms 4.5, 4.6 and 4.7 into Equation 4.3 yields the equation seen in Equation 4.8. This can be expressed equivalently as the inner product of the vector $\mathbf{x} - \mathbf{y}$ with itself (Equation 4.9), which, as defined in Equation 4.4 is the squared Euclidean norm of $\mathbf{x} - \mathbf{y}$. Equivalently, this is the squared Euclidean distance between points $\mathbf{x}$ and $\mathbf{y}$.

$$d_\Phi(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{y}, \mathbf{y} \rangle - \langle \mathbf{x} - \mathbf{y}, 2\mathbf{y} \rangle \tag{4.8}$$

$$\begin{aligned} &= \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} (x_i - y_i)(2y_i) \\ &= \sum_{i=1}^{n} (x_i^2 - 2x_i y_i + y_i^2) \\ &= \sum_{i=1}^{n} (x_i - y_i)(x_i - y_i) \\ &= \langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle = \|\mathbf{x} - \mathbf{y}\|^2 \end{aligned} \tag{4.9}$$

Interpreting the M-dimensional vector, which is the output of the embedding function f_θ, as a Euclidean vector holds both intuitive geometrical meaning as well as mathematical meaning. The employment of Bregman divergences, along with the selection of the mean of support points as the class prototype, minimises dissimilarity among support points, resulting in a suitable 'representative' point in $\mathbb{R}^M$ space. Employing the squared Euclidean distance allows calculation of the 'closest' prototype to a query point, which implies that the query point is most likely associated with that specific class.

4.1.2 Episodic Training

Episodic training, proposed by Vinyals et al. in 2016 [203], is a meta-learning training strategy specifically designed for FSL. The core principle of episodic training is to ensure that the training conditions closely resemble the test conditions in FSL. At test time, the model must adapt to new classes based on a small number of labelled examples for those classes. To emulate these test conditions during training time, the training process is organised into *episodes*, where each step represents one such episode. Instead of generating regular data batches, these episodes simulate the test time scenario.

Each episode is formulated as a N-shot K-way task. The value of K can be smaller than the number of unique labels, which offers certain advantages such as reduced learning complexity, faster training time and potentially increased generalisation. However, smaller

values of K might result in reduced class diversity, less separation between classes and reduced task complexity, potentially leading to poor discrimination during testing and inference. The number of shots N, should be the same across both training and test time [177], as this mimics the test time scenario.

During episodic training, the training data labels are first sampled uniformly, with K classes selected, forming a subset of labels L. Each batch constructed for episodic training consists of a *support set* S and a *query set* Q, i.e $B_E = \{S, Q\}$, from the labelled examples, which mirrors the setup during test time. As previously described in Section 4.1.1, S_k and Q_k represent the subsets containing the examples belonging to class $k \in L$ in the support set and query set, respectively. During training, both sets contain labelled feature vectors. As both S and Q contain labelled examples, a subset of each for class k can be represented as follows: $S_k = \{(\mathbf{x}^s_{k,1}, y^s_{k,1}), ..., (\mathbf{x}^s_{k,N}, y^s_{k,N})\}$ where N is the number of examples in the support set (i.e. 'shots') and $Q_k = \{(\mathbf{x}^q_{k,1}, y^q_{k,1}), ..., (\mathbf{x}^q_{k,M}, y^q_{k,M})\}$ where M is the number of examples in the query set for class k, and $\mathbf{x}$ denotes the feature vector.

By structuring batches as episodes/tasks, the model 'learns to learn' from a few labelled examples, aiming to circumvent the overfitting issue encountered in other fine-tuning or few-shot approaches by emulating the test time scenario. Each model update is based on the performance of the model on a specific episode/task. The support set is employed to create class prototypes, given the model's current state, and the model parameters are updated based on the classification loss between the prototypes and the query instances.

4.1.3 Loss Function

Given the distribution over classes for a query point, formulated in Equation 4.2 and the episodic batch sampling strategy, learning in a prototypical network proceeds by minimising the negative log-probability that a query point $\mathbf{x}$ belongs to the true class k. This loss function is expressed formally in Equation 4.10.

$$L(\theta) = -\log\big(p_\theta(y = k|\mathbf{x})\big) \tag{4.10}$$

Each batch is randomly constructed, so each training episode in an epoch should be unique. Episodes are constructed as described above in Section 4.1.2. A subset of examples in each episode serve to construct the class prototypes, and the loss is calculated for all other points in the episode. The overall loss for an episode is the mean of each query point's loss.

As the probability distribution over classes is based on a softmax over distances to class prototypes, this loss function encourages the network to move query points closer to their correct class prototype. However it has no direct influence on producing tight clusters or increasing distance between prototypes.

4.2 DCASE2021 Few-Shot Bioacoustic Sound Event Detection Challenge

FSL holds great promise as a framework for bioacoustic sound event detection especially in the context of monitoring animal populations using bioacoustics. These monitoring projects often generate extensive, lengthy acoustic data, wherein large portions of the recordings may not contain the events of interest. Manually labelling and segmenting such datasets can be extremely time-consuming, diverting valuable resources away from analysing the relevant data. In contrast, FSL enables users to provide a limited number of exemplar examples of the target event (perhaps extracted from from the beginning of the recording) to the model, which can then identify similar events' onset and offset boundaries. Moreover, the key advantage of FSL lies in its ability to perform well on classes and data not present during training time, either through model weight adaptation or meta-learning. This property is desirable in many applications, including bioacoustics.

As part of their efforts to advance FSL in audio applications, the "Detection and Classification of Acoustic Scenes and Events" (DCASE) community introduced the *Few-Shot Bioacoustic Event Detection Challenge* in 2021 [125]. This challenge has run each year since 2021, as the field of FSL in audio applications advances further. The challenge marks the starting point into this work's investigation into FSL for bioacoustics.

This section provides an overview of the task in Section 4.2.1, followed by a description of

the challenge datasets in Section 4.2.2. Additionally, it presents implementation details of the submitted system in Section 4.2.3 and details of experiments conducted for the 2021 challenge in Section 4.2.4. Subsequently, results and discussion comparing the submitted entry to other challenge entrants, can be found in Section 4.2.5.

4.2.1 Challenge Task

The challenge task is a sound event detection task. An audio recording containing a single class of interest, either a mammal or bird vocalisation, and the onset and offset times for each of the first five sound events in that recording are provided. Using those first five sound events, the task is to identify the onset and offset times for all other sound events belonging to that class within the recording. This can be achieved by classifying slices of the recording as either containing activity related to the class of interest or not, and then inferring the onset and offset times of activity based on these classifications. For more detailed information on this method, please refer to Section 4.2.3.

It is important to note the usage of external models and datasets required prior approval of the task coordinators. Certain external datasets, such as AudioSet [60] and ESC50 [144], and models like ECAPA-TDNN [43], were approved from the beginning of the challenge. The task coordinators also provide two baseline systems for entrants to compare with when developing their own entries. These two baseline systems were:

- **Template Matching** - This approach utilises a spectrogram based cross-correlation technique with normalised cross-correlation to identify instances of templates in a spectrogram. These templates used in this method are derived from the support set. Cross-correlation is computed along the time-axis for each template in the support set. Subsequently, peak-picking is performed to determine the centre of a matched event, and its length based on the template's length.

- **Prototypical Network** - This system represents a more modern, deep learning based approach to bioacoustics analysis. The system is based on the work of Snell et al. [177], which is detailed in Section 4.1 above. The submitted system is also a protonet based system. Additional details of this system can be found in Section 4.2.3.

Regarding the testing scenario, it is a binary classification task requiring generalisation to classes unseen during training time using a FSL framework. This binary classification task entails two labels: positive, indicating the detection of the event of interest, and negative, indicating anything else. During training, the model is trained in a multiclass fashion. While during the testing and validation phases, the model serves as a binary classifier to make the final predictions on activity detection.

4.2.2 Challenge Data

The challenge organisers provide a development set[1] for the task, which is split into training and validation sets. The training set consists of four sets of data from different sources, and the validation set consists of two sets of data from different sources. This section provides an overview of the training, validation and evaluation sets. There is no overlap between the training set and validation set classes. The evaluation dataset used to rank the performance of entries was released after the completion of the challenge[2].

The challenge dataset comprises a diverse range of bioacoustic audio, encompassing not only bird sounds but also other types of bioacoustic data. While the core focus of this PhD research revolves around tasks involving bird audio, the decision to participate in this challenge, which centres on FSL in bioacoustics, was made due to its relevance to the specific domain of automated bird monitoring through vocalisations.

Training Data

The training set is comprised of four sources, these are also summarised in Table 4.1:

- **BV** - A dataset containing five audio files containing bird vocalisations, recorded from four locations in the state of New York. Each recording is two hours long and are all recorded using the same hardware, the Cornell Lab of Ornithology's Recording and Observing Bird Identification Node [163]. The dataset has been annotated for the presence of flight calls from 11 different species of bird. The average call duration is approximately 150 ms with estimated fundamental frequencies between 2 kHz and 10

[1] https://doi.org/10.5281/zenodo.4543504
[2] https://doi.org/10.5281/zenodo.4864755

Overall	**Number of Audio Recordings**	11
	Total Duration	14h20m
	Total Classes	19
	Total Positive Events	4686
BV	**Number of Audio Recordings**	5
	Total Duration	10h
	Total Classes	11
	Total Positive Events	4686
	Sampling Rate	24 kHz
HT	**Number of Audio Recordings**	3
	Total Duration	3h
	Total Classes	3
	Total Positive Events	435
	Sampling Rate	6 kHz
JD	**Number of Audio Recordings**	1
	Total Duration	10m
	Total Classes	1
	Total Positive Events	355
	Sampling Rate	22.05 kHz
MT	**Number of Audio Recordings**	2
	Total Duration	1hr10m
	Total Classes	4
	Total Positive Events	1234
	Sampling Rate	8 kHz

Table 4.1: Details of the DCASE2021 Few-Shot Learning Task training dataset. This table includes overall statistics about the training dataset, as well as details on the subsets which comprise the training set. Positive event means an event matching one of the labels.

kHz. There are 2662 labelled positive events in the dataset and the audio was recorded at a sampling rate of 24 kHz.

- **HT** - A dataset of hyena vocalisations recorded in Kenya as part of a project on multi-species communication and behaviour. The dataset contains three audio recordings and has a duration of 3 hours. The vocalisations were recorded using custom GPS/acoustic collars deployed on female hyenas. This dataset contains 3 classes referring to different vocalisation types and there are 435 positive events in the dataset. The audio in this dataset was recorded at a sampling rate of 6 kHz.

- **JD** - A single recording of one male jackdaw during breeding season, recorded in

Seewiesen, Germany. This recording is 10 minutes in duration and recorded using a small 'backpack'. The dataset has one class, the presence/absence of the jackdaw vocalising and there are 355 positive events. This audio was recorded with a sampling rate of 22.05 kHz.

- **MT** - A dataset of meerkat vocalisations recorded at the Kuruman River Reserve, South Africa. This dataset contains two audio recordings with a total duration of 1 hour 10 minutes. This dataset comes from the same project as HT, mentioned above, and made use of similar GPS/acoustic collars. The dataset contains 4 classes, also referring to different vocalisation types and there are 1234 positive events in the dataset. The audio in this dataset was recorded at a sampling rate of 8 kHz.

Training set annotations are multiclass. Each entry in the annotation file contains the audio filename, the onset and offset times in seconds and positive (POS), negative (NEG) and unknown (UNK) values for each class. UNK indicates uncertainty and for the purposes of the challenge, are treated as the same as NEG.

Validation Data

The validation set is comprised of two sources and is summarised in Table 4.2:

- **HV** - This dataset is taken from the same project as HT in the training set. They are recorded using the same equipment as HT and in the same location There is no overlap between the vocalisations in the two datasets. The audio in this dataset is also recorded at a sampling rate of 6 kHz.

- **PB** - A dataset of six, 30 minute bird vocalisations recorded along the Polish Baltic Sea coast. Recordings were made in 2016, 2017 and 2018 during migration season and were all recorded using the same hardware. Each recording contains only one target class; three recordings target song thrush calls, and the other three recordings target blackbird calls. The audio in this dataset is recorded at a sampling rate of 44.1 kHz.

Validation set annotations are binary, however, the class distribution is heavily imbalanced and most segments of audio belong to the negative class. Each entry in the annotation file

Overall	**Number of Audio Recordings**	8
	Total Duration	5h
	Total Classes	2 (Binary)
	Total Positive Events	310
HV	**Number of Audio Recordings**	2
	Total Duration	2h
	Total Positive Events	50
	Sampling Rate	6 kHz
PB	**Number of Audio Recordings**	6
	Total Duration	3h
	Total Positive Events	260
	Sampling Rate	44.1 kHz

Table 4.2: Details of the DCASE2021 Few-Shot Learning Task validation dataset. This table includes overall statistics about the dataset, as well as details on the subsets within the set. Positive event means an event matching one of the labels.

contains the audio filename, the onset and offset times in seconds and positive (POS), negative (NEG) and unknown (UNK) values for each class. UNK indicates uncertainty and for the purposes of the challenge, are treated as the same as NEG. When constructing the support set using validation or testing data, the first 5 positive events are used. As these are audio events of varying duration, the first 5 positive events may result in a different amount of support audio.

Evaluation Data

After the challenge deadline, details and ground truth labels for the evaluation dataset were released by the challenge organisers. It comprises of three sources and is summarised in Table 4.3:

- **DC** - A dataset of bird vocalisations produced during the *dawn chorus*. The vocalisations are taken from the Dawn Chorus Project, a worldwide citizen science project. In particular the vocalisations used in this dataset were recorded on Zoom H2 recorders, at three locations in southern Germany. The dataset contains 12 recordings of varying length, and each recording contains one of three target species. There are 1192 positive events across all files. The audio in this dataset is recorded at a sampling

Overall	**Number of Audio Recordings**	31
	Total Duration	2h 20m
	Total Classes	2 (Binary)
	Total Positive Events	2302
DC	**Number of Audio Recordings**	12
	Total Duration	1h 41m
	Total Positive Events	1192
	Sampling Rate	44.1 kHz
ME	**Number of Audio Recordings**	2
	Total Duration	19m
	Total Positive Events	73
	Sampling Rate	48 kHz
ML	**Number of Audio Recordings**	17
	Total Duration	20m
	Total Positive Events	1037
	Sampling Rate	44.1 kHz

Table 4.3: Details of the DCASE2021 Few-Shot Learning Task evaluation dataset. This table includes overall statistics about the dataset, as well as details on the subsets within the set. Positive event means an event matching one of the labels.

rate of 44.1 kHz.

- **ME** - This dataset is taken from the same project as MT in the training set. It contains meerkat vocalisations recorded at the Kuruman River Reserve in South Africa. Unlike MT, the recordings in ME are of one meerkat followed using a directional microphone at a distance of less than 1m. The dataset contains 2 recordings, with 73 positive events across the two files. They are also recorded at a sampling rate of 48 kHz, as opposed to the 8 kHz recordings present in MT.

- **ML** - A dataset of mammal and bird vocalisations curated from the Macaulay Library, a digital archive maintained by the Cornell Lab of Ornithology. This dataset contains 17 recordings, each labelled for vocalisation activity from one species. The dataset contains 14 mammals (not including hyena or meerkat) and 3 birds (not including songbirds). Each recording contains only one target class. There are 1037 positive events across the entire dataset. The audio in this dataset is recorded at a sampling rate of 44.1 kHz.

4.2.3 Implementation

Data Preparation

In both the development and evaluation sets, all audio files undergo a series of preprocessing steps. Firstly, they are resampled to 22050 Hz and normalised to -2 dBFS. Subsequently, they are transformed into mel spectrograms. Considering the diverse range of species present in the data, no band-pass filtering is applied to the audio signals before the transformation into a TF representation. This decision was made to enable the network to generalise effectively to new, unseen classes after training, as band-pass filtering could potentially eliminate valuable information relevant to classification. The mel spectrogram transformation is configured to use 128 mel bins, a FFT size of 1024 samples, and a hop of 256 samples, which aligns with the configuration used in the challenge's baseline systems.

Given the high noise content and varying dynamic range present in wildlife field recordings, certain experiments utilise a fixed parameter PCEN [211] transformation. PCEN has been proposed as a method of improving robustness to channel distortion in keyword spotting tasks and is discussed in detail in Section 5.2.3, where various modern learnable frontends are evaluated on a shared bioacoustics task. For the purposes of this chapter, the formulation of PCEN is shown in Equation 4.11.

$$\text{PCEN}(t,f) = \left(\frac{E(t,f)}{(M(t,f)+\epsilon)^{\alpha}} + \delta\right)^{r} - \delta^{r} \tag{4.11}$$

$$M(t,f) = (1-s)M(t-1,f) + sE(t,f) \tag{4.12}$$

PCEN's input is a TF representation $E(t,f)$ and it consists of two operations: Automatic Gain Control (AGC) and Dynamic Range Compression (DRC). The smoothed spectrogram $M(t,f)$ (Equation 4.12, smoothing implemented through an IIR filter parameterised by s), serves as an estimate of the background noise. AGC is performed through division of $E(t,f)$ by $M(t,f)$ raised to the power of α, with an epsilon value to prevent division by zero. This process emphasises changes relative to the recent spectral history along the temporal

axis [115]. The DRC is governed by the parameters (δ, r). While PCEN can be made trainable, in these experiments, fixed values shared across all frequency bins are used. Similar to the reasoning for not applying band-pass filtering, setting PCEN's parameters to suit all examples in the dataset proves challenging due to the dataset's varied and multi-species audio content. The fixed parameter values are based on the guidelines set out in [115].

PCEN is employed to reduce noise in the spectrogram and provide normalisation, gain control and compression to the audio's TF representation. Although PCEN does reduce the noise present in the spectrogram, non-stationary sources of noise still pose an issue and may affect the features.

Once the spectrogram has been calculated for an audio file, it is divided into segments, each associated with the relevant class label. These segments are set to be 200 ms long, with a hop of 50 ms. Considering these segment hyperparameters coupled with the spectrogram hyperparameters, the input feature $\mathbf{x}$ to the protonet becomes a 2-dimensional TF representation of 17 time frames and 128 frequency bins, i.e. $\mathbf{x} \in \mathbb{R}^{17\times128}$. In the following experiments, the protonet systems are trained and tested using both log-mel spectrograms and PCEN-mel spectrograms.

Data Augmentation

To enhance the model's performance and create an embedding space capable of generalising to loss of information, temporal and frequency shifts, as well as new unseen classes, various data augmentation techniques are employed on the extracted spectrogram representations utilised in system development. These data augmentations draw inspiration from the methodologies presented by Park et al. [138] in SpecAugment and Mario Lasseck [98] in their technical report on a previous DCASE bird audio detection challenge. These augmentations are implemented using PyTorch's Audio toolset, Torchaudio [215], which incorporates many of the techniques specified in [138]. The following augmentations are applied to the development data:

- **Time stretching** - A time stretch is applied to each spectrogram, resulting in either its shortening or lengthening without any change in pitch info. This causes a warping

along the horizontal axis of the spectrogram based on a specified factor. For these experiments, a stretching factor of 5% is chosen, leading to augmented spectrograms at approximately 0.95x and 1.05x the playback speed of the original audio file.

- **Time masking** - Time masking is implemented by masking T consecutive time steps, with all frequency information set to 0 within that interval. The starting time step and interval are randomly chosen from a uniform distribution, following Torchaudio's implementation. This masking is applied to each 5s chunk of audio before reconstructing it as one spectrogram during dataset construction.
- **Frequency masking** - Similar to time masking, frequency masking applies augmentation along the vertical, or frequency axis of the spectrogram. It involves random, uniform sampling to determine the frequency to begin masking from and the number of consecutive frequency bins to mask. To increase the variability in the augmented data, this augmentation is also applied in 5s chunks.

The data augmentation artificially increase the amount of data in the support set, effectively creating additional 'shots' for the FSL problem. These augmentations also help the system handle various variations present in the data and enable better classification performance in the face of unseen classes and other challenges. By applying these data augmentation techniques, the model's ability to generalise to different scenarios and improve performance is increased.

Model Architecture and Training

The architecture of the encoder, serving as the embedding function f_θ, is a simple convolutional architecture, illustrated in Table 4.4. This architecture closely resembles the baseline protonet [125], employing convolutional blocks of 128 filters. However, it differs by featuring a reduced network depth and additional pooling after the convolutional layers to achieve the target number of dimensions in the output vector. This reduces model complexity and adheres to the guiding principle of using low-resource classifiers throughout this thesis.

This architecture takes the input TF feature, $\mathbf{X} \in \mathbb{R}^{17\times128}$, and transforms it to a 128-dimensional vector, $f_\theta : \mathbf{X} \rightarrow \mathbb{R}^{128}$. Subsequently, this 128-dimensional vector is utilised

Encoder Architecture	
Layer 1	ConvBlock128
Layer 2	ConvBlock128
Layer 3	ConvBlock128
Layer 4	Flatten/Reduce (128 dims)
ConvBlock128 Architecture	
Sub-Layer 1	Conv2D(128 filters, kernel_size = (3, 3))
Sub-Layer 2	BatchNorm
Sub-Layer 3	ReLU
Sub-Layer 4	MaxPool2D(pool_size = (2, 2))

Table 4.4: Encoder architecture for the submitted prototypical network and ConvBlock architecture. The input to the network f_θ is a TF-representation of a segment of audio $\mathbf{X} \in \mathbb{R}^{T \times F}$, which is transformed to a 128-dimensional vector, i.e. $f_\theta : \mathbf{X} \to \mathbb{R}^{128}$.

to determine $p(y = k|\mathbf{X})$ using Equation 4.2, and consequently the classification of the feature vector $\mathbf{X}$.

Training of the prototypical network involves minimising the negative log-probability of the vector belonging to its correct class. This is achieved using the SGD algorithm with an initial learning rate of 0.01 and a momentum factor of 0.85. To improve the training process, the learning rate is scheduled to halve when a plateau is reached, with a patience of 5 epochs and a threshold of 0.01. Based on our experiments, this configuration provided a favourable training environment. All models are trained for 150 epochs, although the best-performing model achieved the minimum loss at epoch 126. Random Over-Sampling is employed to address the issue of class imbalance during training, implemented using the Python *imblearn* package [101]. This is necessary due to the significant skew in the class distribution of the data.

Using the episodic batch sampling strategy discussed above in Section 4.1.2, the protonet is trained as a 10-way 5-shot system with the number of shots fixed at 5 to mirror the test time conditions faced by the network. The system is trained using $K = 10$ ways to facilitate better generalisation, even though the inference time use case is binary classification. Using a small K may result in an embedding function that does not generalise well. This aligns with the findings of Wang et al. [210], who hypothesise that training a model in a binary

"one-vs-all" scenario hinders its ability to generalise to the 'all' scenario. To create each episode, we randomly generate support and query sets from the available data. Prototypes are computed from the support samples, and the loss is then computed based on the classification of the query points, as calculated using Equation 4.10. The aim of this training process is to optimise the prototypical network to effectively handle the FSL task and achieve robust performance in the challenge.

Post-Processing and Model Evaluation

During the testing and validation phases, support samples are fed into the model and a prototype representation is calculated. It is important to note that the spectrogram segments have a duration of 200 ms with a hop of 50 ms. Given that the first five audio events form the support set, it is worth noting that the support set may consist of more than five data points. In the context of audio, the term 'shots' pertains to labelled events, each potentially varying in duration. Consequently, this variability might lead to a scenario where more than N spectrograms represent N events, as elaborated in Section 2.3.2. This raises a consideration related to the "training conditions should match test conditions" aspect of episodic training. Moreover, since the negative class encompasses all other audio content in the recording, there is an abundance of support points for the negative class. Given that the evaluation follows a "one-vs-all" binary classification scenario, restricting the number of support samples for the negative class to 5 may not allow for the creation of a suitable class prototype. In these experiments, 50 examples from the negative class's support set are chosen to calculate the negative class prototype. This is chosen as a compromise between not having enough support points to represent the negative class, and using many more points than exist in the support set.

Evaluation of the model involves calculating the probability of a query point belonging to the positive class. As shown in Section 4.1.1 and Equation 4.2, this is accomplished through a softmax operation on the distances between query points and class prototypes. The chosen class corresponds to the one with the highest probability, indicating the likelihood of the query point belonging to that specific class.

In the context of binary event detection, probability thresholding is applied. The probability threshold is set at 0.5, though its value could be adjusted based on the application's requirements and the relative importance of recall versus precision.

For an audio recording, this classification process results in labeling each 200 ms segment as either positive or negative. With a feature hop of 50 ms, each 50 ms audio interval is designated as containing the class of interest or not. Since individual segments are classified without temporal modeling, a median filtering procedure with a window length of 5 samples is employed to mitigate the effect of abrupt transitions that might occur due to incorrect segment classification.

To calculate onset and offset times from the frames, a 1D edge detection operation is applied using the kernel defined in Equation 4.13. This kernel is convolved (as shown in Equation 4.14) with the classification results after the median filtering stage. The outcome of this convolution determines the points where events begin (onset), indicated by a convolution output of 1, and where they end (offset), marked by a convolution output of -1.

$$k[n] = [1, -1] \tag{4.13}$$

$$(y * k)[n] = y[n] - y[n-1] \tag{4.14}$$

By utilising the indices of these values and considering the feature hop size (50 ms), the onset and offset times are calculated. Subsequently, a post-processing technique endorsed by the challenge organisers in their deep learning baseline is employed to refine these onset and offset times. Any positive instances shorter than 60% of the shortest positive labelled instance are eliminated from the final results. The evaluation demonstrates that this post-processing step reduces false positive events with little change in the amount of true positives.

4.2.4 Experimental Results

Performance is evaluated on the validation data, provided in the development set released as part of the challenge. To assess the FSL model, the challenge organisers provided evaluation code[3] which draws inspiration from the *mir_eval* project [150]. This code calculates an event-based, macro-averaged F1-score across all data sources, which is employed as the evaluation metric. The F1-score, a measure of the model's precision and recall trade-off, for each audio file is calculated as the harmonic mean of precision and recall. The mean of these individual F1-scores is then computed to obtain the macro-averaged F1-score reported. This is achieved by calculating the Intersection over Union of the predicted event onset and offset times with the provided ground truth predictions. Since predictions are generated following the first five positive events, the evaluation process exclusively considers events occurring after the offset of the fifth positive event. The evaluation code further generates confusion matrices for each file and computes precision, recall, and the F1-score per file.

It is important to highlight that the distribution of positive and negative events within the evaluation set is significantly skewed toward the negative class. The specific distribution varies across audio files, but for context, in the DC data, positive events occur approximately 30% of the time, in the ME data, they occur approximately 1% of the time, and in the ML data, they occur approximately 18% of the time.

Given the resemblance between the proposed system and the baseline, the challenge baseline results were reproduced to facilitate comparison with the proposed system, using the code provided by the challenge organisers. Despite the challenge description indicating a baseline F1-score of 0.415, these results could not be replicated; the reproduced results attained a maximum F1-score of 0.304. Results in this section are also reported using our system, separately for log-mel spectrograms only, followed by the application of data augmentation, and subsequently, data augmentation along with PCEN.

Table 4.5 presents the performance of each system on the validation dataset, reporting F1-score, precision and recall across the entire dataset. From this we can see that the

[3]Evaluation metric code for the DCASE Few-Shot Bioacoustic Detection Challenge is available at `https://github.com/c4dm/dcase-few-shot-bioacoustic/tree/main/evaluation_metrics`

Model	F1-Score	Precision	Recall
Baseline Reproduction	**0.304**	**0.456**	0.228
(A) Log-mel Spectrograms	0.166	0.108	0.355
(B) Data Augmentation	0.205	0.143	0.360
(C) Data Aug. + PCEN	0.262	0.200	**0.381**

Table 4.5: Results of the baseline protonet from the challenge organisers, our protonet system (A), our system plus data augmentation (B), and our system plus data augmentation and PCEN features (C) experiments on the validation dataset. Best results in **bold**.

baseline reproduction outperforms our proposed systems on this data.

Comparing the baseline system to our proposed systems as a group, it becomes apparent that the baseline system prioritises precision over recall. The baseline system demonstrates superior precision but inferior recall compared to any of our proposed systems. Whenever the baseline system predicts a positive event, it tends to be more accurate than the positive event predictions from the other systems. However, due to the baseline reproduction's lower recall, it misses more of the positive events and has a higher count of false negatives compared to our proposed systems.

It can also be seen in Table 4.5 when comparing systems A and B that the introduction of data augmentation to our system during training leads to an enhancement in the F1-score. This enhancement primarily stems from improved precision in System B (with a gain of 0.035 over System A), while the increase in recall remains marginal (improved by 0.005 over System A). Introducing data augmentation to the feature extraction pipeline allows the model to more accurately identify positive events, reducing the occurrence of false positives in the results while maintaining the same level of true positive event detection. However, no further studies have been conducted to identify the specific data augmentation methods responsible for the observed increase in performance. Findings presented in [138] suggest that all three augmentation methods have an impact, with time/frequency masking playing the most prominent role in performance improvement.

The results of adding PCEN to the feature extraction stage are observed in System C. Here, a noticeable increase in the F1-score is evident across both subsets of the validation dataset. The improvement is primarily seen in the precision of the system, an improvement of 0.057 in

Data Subset	F1-Score	Precision	Recall
PB	0.222	0.201	0.249
HV	0.320	0.199	0.825

Table 4.6: Breakdown of results using Data Augmentation and PCEN by validation data subset.

System C over System B, which is an overall improvement of 0.092 over System A. Furthermore, a more substantial improvement in recall is noted, with a gain of 0.021 over System B and a gain of 0.026 over System A. The application of PCEN outperforms traditional logarithmic compression, underlining the superiority of PCEN in handling noisy data, as suggested in [115]. Notably, the PCEN implementation used in these experiments involved fixed parameters rather than fully learnable per-channel parameters. Further investigation using learnable PCEN will be seen in Chapter 5.

System C is the best performing of the proposed systems and predictions from this model were used as the entry to the challenge. In comparison to the baseline reproduction, there were significant improvements to recall, with the system capable of identifying 38% of positive events within the validation set. However, this comes at the expense of precision, as a number of false positives are observed in the predicted output. The integration of PCEN features over traditional log-mel spectrograms proved pivotal to this improvement, lending credence to claims in [117] that PCEN can significantly outperform log-mel scaling when used on noisy data, with no significant increase in computational complexity. Table 4.6 provides a breakdown of system C's performance by subset in the validation set. On both datasets, recall is higher than precision, with recall on the *HV* dataset being much higher at 0.825. This could be attributed to the *HV* dataset containing less events less often (45 positive events to classify, 2 hour duration) than the *PB* dataset (255 positive events to classify, 3 hour duration).

From the overall results in Table 4.5 and the results by subset in Table 4.6, it is apparent that the model is prone to false positives while still permitting a considerable number of false negatives, particularly evident in the *PB* dataset. The model is a poor discriminator between the classes, and although an embedding space has been learned according to the training

procedure, it is a poor performer on this task. Despite the advancements introduced within the feature extraction pipeline, the system entered into the challenge did not surpass the baseline (as stated in either the challenge description or the reproduced results) on the validation dataset. In the following section (Section 4.2.5), the results of the challenge will be discussed alongside a brief overview of some other entries to the challenge, as well as comparing the performance and model complexity of those entries relative to this entry.

4.2.5 Challenge Results

The 2021 edition of the DCASE Few-Shot Bioacoustic Event Detection Challenge had 25 entries, including the baseline systems, from 8 teams, including the challenge organisers themselves. The evaluation process utilised the same event-based, macro-averaged F1-score employed during the validation phase. The entry detailed above achieved an F1-score of 0.350, with a confidence interval of ± 0.020. This placed the entry 5th among 26 entries in the challenge, and in the team ranking was placed 3rd. The overall rankings of the challenge can be found in Table 4.7, this table contains both baselines proposed by the challenge organisers. It is worth highlighting the gap in performance between submissions below the fourth place position. Also of interest is that the F1-score on the validation set does not precisely mirror the overall ranking exhibited by the evaluation set. The Johannsmeier [82] system excelled on the validation set, yet it ranked second-to-last in the overall standings. Conversely, the template matching baseline performed poorly on the validation set but exhibited a comparable overall performance to our submission on the evaluation set, and the protonet baseline performs worse despite an F1-score of 0.415 on the validation set. Within Section 4.2.4, our system was unable to outperform the prototypical network on the validation set (where the baseline reproduction achieved an F1-score of 0.304, whereas our protonet system attained an F1-score of 0.262). However, it is important to note that this difference in performance is not accurately reflected in the final system rankings, where our system surpasses both the protonet baseline and the template matching baseline.

While the assessment results only considered the F1-score on the evaluation dataset, the challenge organisers aimed to gather insights regarding architecture types, employed FSL frameworks, and model complexity. Among the submissions, our system was the least

Rank	Team	Eval. F1	Val. F1	DC F1	ME F1	ML F1
1	Zou [214]	$\mathbf{0.384 \pm 0.022}$	0.553	0.206	0.680	0.673
2	Tang [192]	0.383 ± 0.022	0.514	0.256	0.615	0.433
3	Anderson (Ours) [2]	0.350 ± 0.020	0.262	0.199	0.566	0.568
4	Template Match Baseline [125]	0.348 ± 0.022	0.020	$\mathbf{0.322}$	0.471	0.295
5	Cheng [33]	0.238 ± 0.019	0.463	0.106	0.535	$\mathbf{0.788}$
6	Protonet Baseline [125]	0.201 ± 0.019	0.415	0.085	$\mathbf{0.727}$	0.557
7	Zhang [220]	0.168 ± 0.013	0.544	0.081	0.451	0.299
8	Johannsmeier [82]	0.152 ± 0.015	$\mathbf{0.586}$	0.065	0.643	0.358
9	Bielecki [14]	0.084 ± 0.013	0.518	0.031	0.563	0.514

Table 4.7: DCASE 2021 Few-Shot Bioacoustic Event Detection Challenge team rankings, as reported by the challenge organisers [125]. This table contains each team's best submission and includes the results on the challenge evaluation set, the validation set used in training, and the results per data source in the evaluation set (DC: Dawn Chorus, ME: Meerkat, ML: Macaulay Library). Best Results for each in **bold**.

complex out of all submissions, comprising a model with 132k parameters and a straightforward prediction mechanism relying on squared Euclidean distance (typical in Prototypical Networks). The system closest in complexity with better performance is the submission by Tang et al. [192], which leveraged a ResNet [71] based system combined with Embedding Propagation [159]. Given that the submissions by Yang et al. [214] and Tang et al. [192] outperformed the implementation outlined in this chapter and significantly outperformed the remaining entries, a short discussion about these submissions in relation to ours is appropriate.

The top submission from Yang et al. [214] (referred to as 'Zou' in Table 4.7) employs a transductive inference-based approach to FSL, contrasting with the meta-learning strategy used by protonets. This submission achieved an F1-score of 0.384. The authors drew inspiration from TIM, proposed by Boudiaf et al. [17] in 2020. TIM, a modular framework suitable for use with various architectures, initially trains the model without meta-learning techniques to establish a base feature extractor, denoted as f_θ. For each few-shot task, a new classifier is trained, and the query set elements are assigned labels according to the current state of the model (creating pseudo-labels for the query set). Subsequently, the model is refined using both the labelled support set and the pseudo-labelled query set. This refinement involves cross-entropy loss with labelled support samples and maximising mutual information between query samples and their pseudo-labels for the query set. Both [17, 214]

have found that keeping the feature extractor parameters fixed yields optimal performance. Yang et al. also employed the Kullback-Leibler Divergence, guided by the further work of Boudiaf et al. [16], to predict the positive and negative event proportions based on predictions from previous iterations.

Tang et al.'s [192] top-performing submission utilises the protonet framework and a meta-learning system to tackle the FSL challenge. This submission obtained an F1-score of 0.383. Their best-performing entry employs a readily available model, specifically ResNet12 [71], trained on the 'animal' subset of AudioSet [60], as a feature extractor. They incorporate embedding propagation [159] to enhance the model's generalisation capacity towards unseen classes. Embedding propagation, an unsupervised, non-parametric regularisation technique, produces interpolated points from network output vectors, each point representing a weighted sum of its neighbours. This technique effectively eliminates undesirable noise from feature vectors and enhances prototype construction. Similar to our approach, this system also integrates static PCEN during feature extraction.

The difference in performance between the two systems is marginal. However, when comparing their performance on individual datasets in Table 4.7, it is apparent that the Zou submission outperforms the Tang submission on all datasets except for the dawn chorus dataset (DC). The Zou submission is also less complex than the Tang submission, with significantly fewer parameters (468k parameters in the Zou system compared to 4M parameters in the Tang system). Nonetheless, the Zou system's approach requires additional training steps per task to train the classifier and undergoes some generality loss through fine-tuning. On the other hand, the Tang system, being protonet-based, requires no additional training, and inference relies solely on the squared Euclidean distance between query points and class prototypes, an attractive trait for protonets as new task classification does not require additional training. Both systems offer distinct advantages, and the choice between them depends on factors such as memory constraints and the acceptability of classifier fine-tuning. When contrasting these two systems with our entry, their performance stems from advanced feature extraction and regularisation (Tang) or information maximisation (Zou), while our system stands apart through distinct feature extraction and data augmentation strategies. These modifications contribute to enhanced robustness and

generalisation within the learned embedding space when handling limited data scenarios, as encountered in FSL scenarios.

Following the challenge's conclusion, further investigation and analysis of the submitted system was undertaken, in order to inform what modifications could be made to increase performance further. This was done with a view to equal or better the performance to the top entry, while maintaining the low model complexity of the submitted system.

4.3 Investigation following the DCASE2021 Challenge

After the conclusion of the challenge, further analysis of the submitted entry was undertaken with the aim of identifying potential areas for improvement. Given that a protonet learns a complex, non-linear mapping of a feature vector from a high-dimensional space R^D to a lower-dimensional embedding space in R^M, and performs classification via the squared Euclidean distance between a query point and a class prototype, an in-depth investigation into the properties and characteristics of this embedding space provides insight into this mapping and the protonets ability to discriminate between positive and negative events. This, in turn, can allow for potential improvements to be made, such as architectural adjustments, innovative training methodologies, or modifications to the feature extraction pipeline.

In the following sections, analysis of the network is undertaken, uncovering certain issues that have been identified as contributors to the sub-optimal results. Subsequently, modifications to the system aiming to address these issues are proposed and evaluated. This is motivated by the goal of refining the system's performance and achieving higher levels of performance while maintaining the system's current simplicity.

4.3.1 Analysis of Embedding Space using t-SNE

The vectors inside the embedding space of the model are analysed using t-Distributed Stochastic Neighbour Embedding (t-SNE) [200]. The embedding space is a 128-dimensional space, which, due to their high dimensionality, cannot be visualised without the aid of a suitable projection technique. t-SNE is a dimensionality reduction and data visualisation technique utilising machine learning, and is a suitable choice for visualisation. Its

fundamental objective is to transform data points from their original high-dimensional space into a lower-dimensional one, preserving the pairwise similarity relationships that exist among these data points in the higher-dimensional space.

t-SNE achieves this objective by constructing two probability distributions, one for the high-dimensional space and another for the low-dimensional space and subsequently minimises the Kullback-Leibler divergence between these distributions. Through this process, t-SNE projects data points from the higher-dimensional embedding space to a two-dimensional visualisation space.

A characteristic of t-SNE is its ability to preserve local structure in the data. Moreover, the balance between preserving local and global structure can be fine-tuned by adjusting the perplexity hyperparameter. it is crucial to recognise that t-SNE, in contrast to other dimensionality reduction techniques such as Principal Component Analysis (PCA) [141], is non-linear. This non-linearity makes t-SNE adept at capturing intricate and complex structures which may not be captured by PCA. PCA (by design) cannot capture local structure, instead it captures the global structure of the data.

For the purposes of visualising the embedding space, the preservation of local structure is desirable. This is because, in the context of the embedding space, data points that are spatially close to each other in the high-dimensional space retain their proximity in the lower-dimensional visualisation space. Global structure preservation is of secondary concern, as the current analysis does not involve comparisons between clusters or structures. The focus here is judging how discriminative the embedding space is, specifically, its ability to neatly cluster points belonging to the same class while minimising the risk of points being inadvertently assigned to other class. This analysis helps in addressing the false positive and false negative issues discussed in Section 4.2.5.

These analyses utilise scikit-learn's [142] implementation of t-SNE. Unlike PCA, t-SNE requires some hyperparameter tuning for optimal results. It was initially thought [200] that t-SNE would be robust to hyperparameter values; however, subsequent research [22, 51] has shown this to not be the case. Whilst implementations strive to offer sensible defaults, optimal results for t-SNE can only be achieved through fine-tuning. Fortunately, many

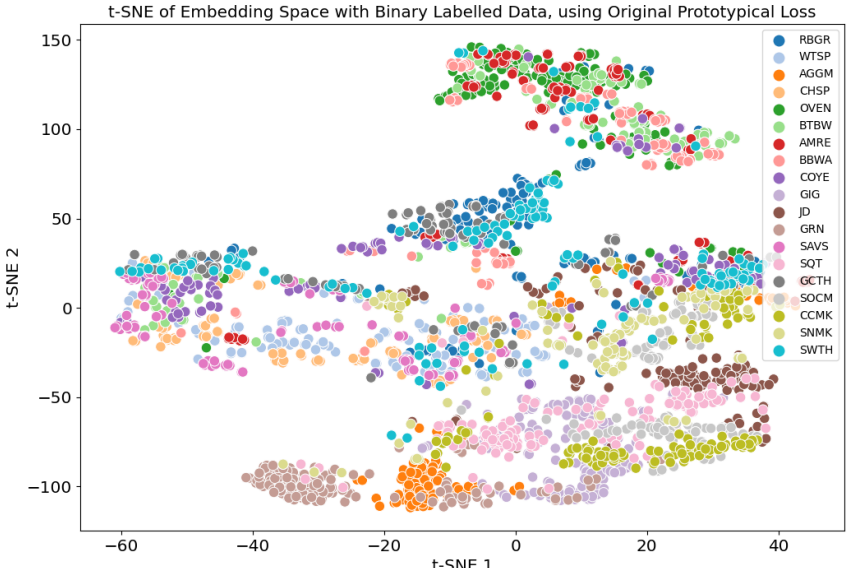

Figure 4.2: t-SNE projection of the embedding space learned by the prototypical network. This figure contains the embeddings of the training data, which is multiclass. 'GIG', 'GRN' and 'SQT' refer to 3 types of Hyena vocalisation from the HT data. 'AGGM', 'CCMK', 'SOCM' and 'SNMK' refer to 4 types of Meerkat vocalisation from the MT data. The remaining classes are 12 different species of bird belonging to the BV and JD data.

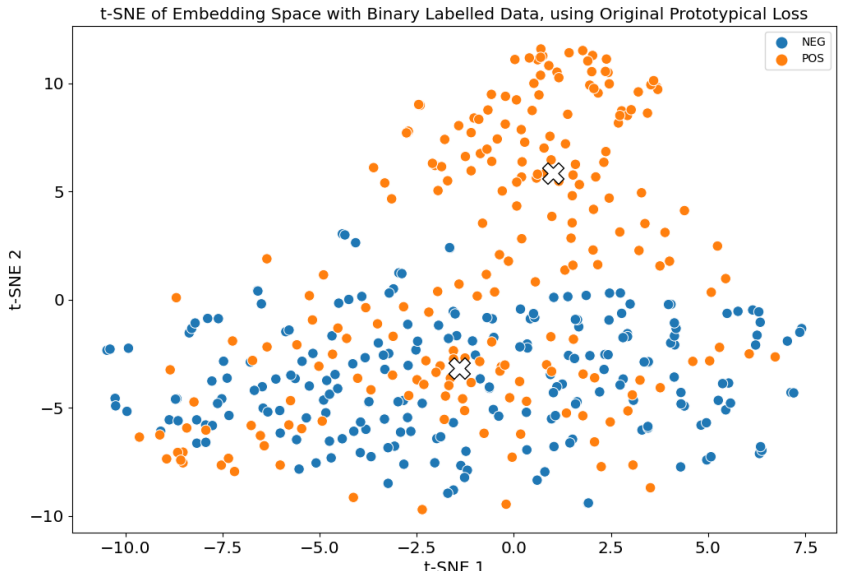

Figure 4.3: t-SNE projection of the embedding space learned by the prototypical network. This figure contains the embeddings of the evaluation data, which is a binary classification problem indicating whether a segment contains bioacoustic activity from the class of interest. Although global structure is not maintained, the prototype representations for both positive and negative classes are marked by a white 'X'.

guidelines exist to streamline the hyperparameter tuning process, potentially minimising or even circumventing the need for extensive tuning.

Given that this work primarily focuses on the analysis of the embedding space rather than its visualisation, the guidelines provided by Gove et al. in [64] are utilised as a starting point. Subsequently, the perplexity hyperparameter is manually tuned, setting it to 30, along with exaggeration $= 1$ and learning rate $= 40$.

Figures 4.2 and 4.3 show the t-SNE visualisation for all the training data (Figure 4.2) and a singular audio recording from the test set (Figure 4.3). These visualisations are of the embeddings produced by the submitted protonet system described above. In all subsequent visualisations, the same file from the test set is used, a recording from the DC dataset containing bird vocalisations during the dawn chorus.

Figure 4.2 shows that, for numerous classes, data points belonging to a specific class tend to coalesce. Nonetheless, some contamination and overlap between classes persist, signifying that, for certain classes, the embedding function is a poor discriminator.

Figure 4.3 offers insight into the embedding space in a binary classification scenario, where the protonet is presented with 5 occurrences of the class of interest to calculate a positive prototype, while a negative prototype is constructed from randomly sub-sampled data from the negative class. These prototypes are denoted by white 'X's in the figure. Every other point belongs to the query set, with colour-coded labels based on their ground truth.

From Figure 4.3, it becomes apparent that the positive and negative classes fail to form distinct clusters. The negative class exhibits more consistent clustering; however, a number of positively labelled query points are located within this cluster. This consistent clustering within the negative class suggests that, although it encompasses all entities apart from the class of interest, the network's learned embedding function is able to generalise to this class. However, the notable absence of substantial distance between the positive and negative classes within the embedding space, despite t-SNE not inherently preserving global structure, indicates minimal separation between these classes. This lack of distance leads to both false positives and false negatives when classifying query points, given the close proximity of class prototypes and a decision boundary that offers limited discrimination between them. This

represents one of the issues brought to light through the visualisation of embeddings.

As mentioned above, many query points belonging to the positive class are located within the negative cluster. This phenomenon can be attributed to various factors: the recording conditions; far-field recordings; or the presence of substantial amounts of noise. Consequently, a degree of similarity exists between these query points and the negative class.

Although PCEN does offer some noise reduction capabilities and dynamic range compression for far-field recordings, it is important to acknowledge that the efficacy of the currently implemented PCEN layer (fixed parameters, non-learnable) limits the amount of and suitability of noise reduction or dynamic range compression. PCEN does yield performance improvements over logarithmically scaled features (as demonstrated in Table 4.5). Nevertheless, the issues surrounding noise and far-field dynamic compression are not entirely mitigated. This issue of noisy data represents the second issue identified by visualisation of the embeddings via t-SNE.

These two issues, query points in close proximity to the negative cluster and the influence of noisy data on embedding quality, lead to sub-optimal clustering and separation of the two classes during test time. In response to these two issues, two potential solutions were explored. The first addresses the lack of distinct clusters by implementing a modified loss function that actively promotes tighter grouping of embedding points originating from the same class while simultaneously maximising separation between distinct classes through the utilisation of a triplet loss function. The second approach aims to combat the detrimental effects of noisy data on embeddings. It involves training the embedding function using multiple representations of the same input feature, diversifying the embedding space. The results of employing these methods are detailed in Section 4.3.4.

4.3.2 Triplet Loss to Improving Clustering and Increase Separation

Triplet loss, first proposed by Schroff et al. in [167], represents a class of loss functions widely employed in systems aiming to group similar items together in an embedding space while keeping dissimilar items further away. This makes triplet loss a prime candidate as a loss function in a prototypical network, which aims to learn an embedding space for use in a

FSL scenario. Triplet loss may then provide a method for refining the embeddings and improving separation between classes during training, ultimately contributing to the development of a more diverse and versatile embedding space.

Triplet loss functions operate by creating triplets of data points, each comprising an *anchor* point $\mathbf{X}_a$, a *positive* point $\mathbf{X}_p$ and a *negative* point $\mathbf{X}_n$. The anchor point signifies the data point for which embedding learning is sought, the positive point corresponds to a data point belonging to the same class as the anchor. In contrast, the negative point belongs to a different class. The objectives of triplet loss functions are twofold: to minimise the distance between the anchor point and the positive point; and to maximise the distance between the anchor point and the negative point. Any pair-wise distance $d(\cdot, \cdot)$ can employed in the triplet loss function. In the context of Prototypical Networks, the squared Euclidean distance is still chosen as the distance function as it is a Bregman divergence (see Section 4.1.1).

Mathematically, the loss function is expressed through Equation 4.15, where the α term represents the margin parameter. α enforces a minimum separation between positive and negative examples and is typically set to a value such as 1. It can also be adjusted as a tunable hyperparameter.

$$L(\mathbf{X}_a, \mathbf{X}_p, \mathbf{X}_n) = max\big(0, d(\mathbf{X}_a, \mathbf{X}_p) - d(\mathbf{X}_a, \mathbf{X}_n) + \alpha\big) \tag{4.15}$$

In contrast to the original prototypical loss function, which seeks to minimise the negative log likelihood of query points belonging to the target class after prototype construction, indirectly encouraging clustering, the triplet loss takes a different approach. It actively adjusts embeddings to promote both inter-cluster separation and intra-cluster cohesion. This approach may lead to enhanced model performance by facilitating the learning of a more complex and diverse embedding function.

A specific formulation of triplet loss suited for Prototypical Networks has been put forward by Doras et al. [48] within the context of few-shot music cover detection. This formulation utilises triplet loss and incorporates semi-hard negative triplet mining. Semi-hard triplet

mining is a strategy where triplets are selected such that the negative point is close to the positive point. This approach provides valuable training information without making the task overly challenging. The formulation of this loss closely resembles the original triplet loss function. However, it employs representations of positive and negative prototypes instead of individual data points. This further improves samples clustering around their respective class prototypes.

4.3.3 Multiple Representations and Introduction of the Background Class

PCEN has the effect of 'whitening' noise [115] and emphasising events with sharp onsets. These attributes are advantageous when representing positive events containing the class of interest. However, when applied to environmental noise present in an audio recording, they may lead to sub-optimal representations of such noise. In contrast, log-mel spectrograms can accentuate the noise floor relative to the signal, and log-scaling lacks the noise-reducing characteristics of PCEN. Consequently, log-mel features are 'noisier' compared to their PCEN counterparts and might offer a more useful representation of the negative class.

To this end, we propose utilising multiple representations during the training process, alongside the introduction of a background class into the training dataset. This newly introduced background class comprises audio segments extracted from all training data files, mirroring the concept of negative features employed during testing. These segments are chosen from sections of recordings that do not contain any positive events. By incorporating both PCEN features and log-mel features, as well as introducing the background class, it is hoped that the learned embedding space will be more diverse. This approach is aimed at further separating the clusters corresponding to positive and negative classes at test time, ultimately improving classification performance during inference.

While it is evident that negative query points cluster together (as seen in Figure 4.3), the presence of positive points within the negative cluster adversely impacts classification performance. Although the PCEN features of the negative class exhibit clustering and are similar to each other, training with a wider range of features and the explicit incorporation of a background class could potentially allow for further separation between the positive and

System	F1-Score	Precision	Recall
Original Submission	0.350	0.488	0.272
Triplet Loss	0.351	0.491	0.273
Mult. + BG Class	0.352	0.494	0.274
Triplet Loss + Mult. + BG Class	**0.368**	**0.522**	**0.284**

Table 4.8: Results on the challenge evaluation set for the originally submitted system, trained using prototypical triplet loss, the system trained using multiple representations, and the system trained using prototypical triplet loss + multiple representations. Best results in **bold**.

negative classes, resulting in enhanced performance.

Although trained using multiple representations, only PCEN features are employed during inference. Determining whether a feature should be represented using PCEN or a log-mel spectrogram a priori is not possible (otherwise, the need for detecting bioacoustic events in the audio would be unnecessary). Given that PCEN has increased overall performance (refer to Table 4.2.5), it serves as the feature of choice during inference.

4.3.4 Results and Discussion

Experiments in this section follow the architecture, training strategy, and post-processing techniques outlined in Section 4.2.3, with modifications to the loss function and adaptations for incorporating multiple feature representations and background class modelling. Evaluation is carried out on the evaluation set, the full details of which were released after the challenge results were made available. The evaluation process and metrics mirror those employed by the challenge organisers, as described in Section 4.2.4. The original submitted protonet system is used as a baseline in the following results.

Table 4.8 provides performance comparisons between our original protonet system and the systems incorporating the aforementioned modifications. As in previous results, the reported metrics are F1-Score, precision, and recall. Table 4.8 can be compared with Table 4.7, as both report performance on the evaluation set.

Comparing the performance across all systems, it is evident that there is only marginal improvement in the overall F1-score when utilising any of the modifications to the initial submission. The improvements when using prototypical triplet loss, or multiple

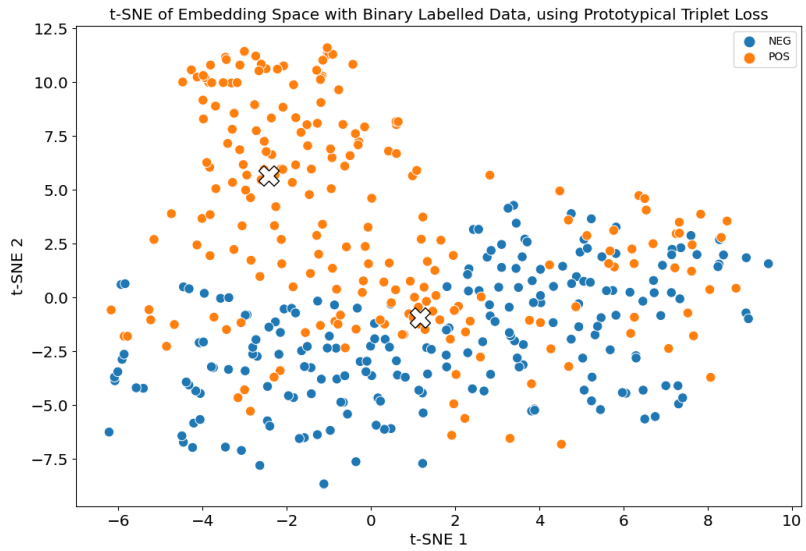

Figure 4.4: t-SNE projection of the the test data embeddings learned by the prototypical network trained using protypical triplet loss. The prototype representations for both positive and negative classes are marked by a white 'X'.

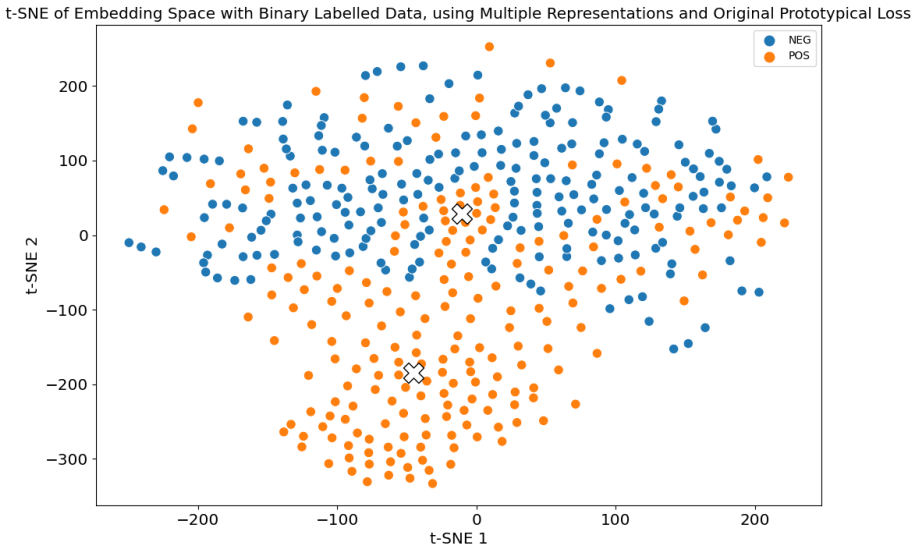

Figure 4.5: t-SNE projection of the the test data embeddings learned by the prototypical network trained using multiple representations and the inclusion of an background class label. The prototype representations for both positive and negative classes are marked by a white 'X'.

representations along with background class modelling in isolation could plausibly be attributed to variances in the non-deterministic training procedure.

Although there is only marginal overall improvement in F1-score with the utilisation of any individual modification, combining both modifications results in a relative improvement of 5% in F1-score, 7% in precision, and 4% in recall. However, these enhancements still fall short of matching the performance achieved by the next highest ranked challenge entry, which attained an F1-score of 0.383 (as seen in Table 4.8). It is important to note that the system remains more lightweight than the next highest entry, and the performance gap has narrowed.

Further analysis of how the modifications impact the model and, consequently the embedding space can be achieved by looking at the embeddings generated by the protonet. Figures 4.4, 4.5 and 4.6 show t-SNE visualisations of protonets trained with prototypical triplet loss, multiple representations plus background class modelling, and both modifications combined, respectively. With reference to these figures and Table 4.8, the effects of each modification individually and combined will be discussed.

For the network trained using prototypical triplet loss, similar to the reported results, there appears to be minimal deviation in the overall structure of the embedding points, as seen in Figure 4.4. While there may be tighter clustering of both classes, many positive query points remain situated away from the primary positive cluster and are closer to the negative cluster. This proximity to the negative cluster results in a higher number of false negatives. The marginal improvement in precision, which may be attributed to a reduction in false positives, can likely be ascribed to the non-deterministic elements of training. Therefore, it can be inferred that the utilisation of prototypical triplet loss in isolation does not lead to performance enhancement of this network on this bioacoustic data.

Moving to the network trained using multiple feature representations and background class modelling, shown in Figure 4.5, there is a some shift in the overall structure, albeit with relatively less separation between the classes. The increase in F1-score is again marginal, with the primary improvement being in precision. As the performance gains are so small, it is still difficult to attribute them to the modification of the system. Similar to prototypical triplet loss, the adoption of multiple feature representations and the modelling of a

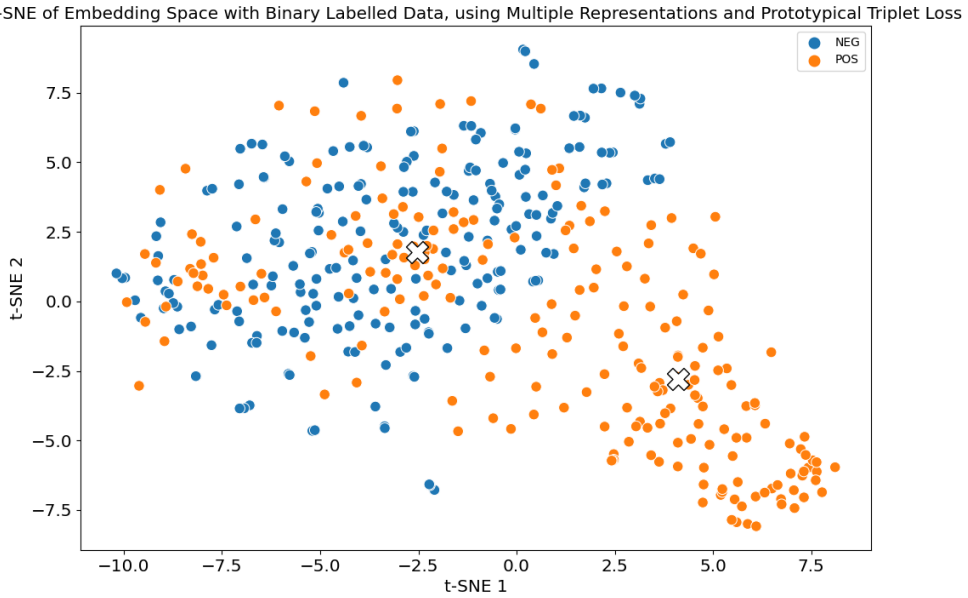

Figure 4.6: t-SNE projection of the the test data embeddings learned by the prototypical network trained using protypical triplet loss plus multiple representations and the background class. The prototype representations for both positive and negative classes are marked by a white 'X'.

background class do not result in performance improvements for this specific model and task.

The usage of prototypical triplet loss in conjunction with multiple feature representations and background class modelling shows the greatest improvement over the original submission, with an F1-score of 0.368. As previously mentioned, this configuration shows a relative improvement of 5% in F1-score, 7% in precision and 4% in recall. While this denotes a step forward from the original system, it remains challenging to confirm its statistical significance without further analysis.

Importantly, this outcome still falls short of surpassing the next highest-ranking entry, which achieved an F1-score of 0.383. Examining the t-SNE visualisation (Figure 4.6) reveals no marked improvement in cluster cohesion or class separation over the original system (Figure 4.3). In comparison to Figure 4.6, it demonstrates a similar degree of separation of separation between positive and negative clusters, especially when contrasted with Figure 4.5, as would be expected using prototypical triplet loss. Overall, the usage of both modifications

may improve the results of this system on this task, but this cannot be confirmed without further statistical analysis. This analysis has not been carried out due to the marginal nature of the improvements. Conducting such an analysis would require additional effort without yielding any improvement or additional insights, and this effort is better focused elsewhere.

Instead, the initial analysis of the submitted systems embedding space strongly suggests that the tendency of query points from the positive class to cluster with those from the negative class can be attributed to the presence of noise and far-field conditions in the recordings. These issues are prevalent in many audio tasks, especially in bioacoustic recordings. Thus, rather than concentrating on incremental refinements to the training regimen or network architecture, effort was concentrated on better feature representations, in particular the possibility of learning better feature representations from the data itself.

The quality of input features is of paramount importance in machine learning, and this extends to FSL, particularly in meta-learning scenarios such as protonets, where models must "learn to learn" and generalise to potentially new and unseen classes at inference time using only a few labelled examples. Bioacoustic audio is marked by challenging recording conditions; recorded in remote environments with omnidirectional microphones, it is particularly susceptible to noise and contains far-field sources, which can detrimentally affect feature quality. Therefore, motivated by the experience of FSL for bioacoustics, the next chapter of this thesis tackles the use of learnable frontends in bioacoustics.

Chapter 5

Learnable Frontends and the Filterbank Initialisation Problem

In speech and audio processing, researchers aim to extract information from audio signals for analysis or to perform a task (such as activity detection). As the techniques used in audio signal processing have advanced, so too have the features extracted from audio signals. This trend continues in the era of deep learning, as systems capable of learning features directly from data have emerged. Chapter 5 presents work evaluating the effectiveness of recent learnable frontends on a bioacoustics task. Additionally, this chapter explores the sensitivity of learnable filterbanks to their initialisation, known as the filterbank initialisation problem. The use of learnable frontends offers potential improvements in performance for tasks using challenging audio, while the promise of learnable filterbanks implies that a system could learn optimal filters for the task, without relying on human-designed scales such as the mel scale.

This thesis chapter is organised as follows: First, a brief introduction is provided in Section 5.1, on the features and representations of audio utilised in non-learnable feature extraction techniques. This will be followed by a discussion on the limitations of these approaches, which learnable frontends seek to overcome. Four learnable frontends, which have been evaluated on a common bioacoustics task, are then described in Section 5.2. The experimental setup is presented, followed by the results and discussion of those experiments

in Section 5.3. The results presented until this point were originally presented [3] in IWAENC 2022. Subsequently, Section 5.4 explores and quantifies the sensitivity of learnable filterbanks to initialisation. Efficient Learnable Audio Frontend (eLEAF) [166, 216] is used to demonstrate this phenomenon, and we investigate the sensitivity of a learnable filterbank to initialisation using various initialisation strategies on two audio tasks: Voice Activity Detection and Bird Species Identification. This work was presented [5] in ICASSP 2023. Further investigation of the filterbank initialisation problem, and suggested mitigation strategies are discussed later in Chapter 6.

5.1 Non-learnable Frontends

Historically, non-learnable frontends, or more precisely non-learnable feature extractors, have been widely used for extracting features from audio data. These frontends operate based on a predetermined set of hyperparameters to extract the relevant features from the audio. A brief description of common extracted features is provided below. In the case of non-learnable approaches, feature extraction is typically performed before the feature is fed into the model, although it is also possible to compute these features during the forward pass of a neural network.

5.1.1 Mel-Frequency Cepstral Coefficients

Until the rise of deep learning, Mel-Frequency Cepstral Coefficients (MFCC) [40] were the most popular extracted feature for acoustic analysis involving speech [91, 128] and also in bioacoustics [65, 184]. MFCCs enable the compression of spectral information into a small number of coefficients. The process involves segmenting the signal into short frames using a windowing function and estimating the spectral density of each frame. This is achieved by applying the Discrete Fourier Transform (DFT) to each frame (Equation 5.1) and passing it through a mel scale based filterbank, inspired by psychoacoustic principles [135, 180]. The mel scale is a perceptual scale where pitches are judged by listeners to be of equal distance from one another. The reference frequency in the mel scale is 1000 Hz (i.e. 1000 mel is equivalent to 1000 Hz). Conversion between frequency f hertz to m mels is described by a

formula from [135] (refer to Equation 5.2, and the inverse expression in Equation 5.3). Taking the logarithm of the output of this M-channel filterbank (denoted as $\log \hat{y}_m, m = 1, \ldots, M$) and passing it through a Discrete Cosine Transform (DCT) (Equation 5.4) decorrelates the features. The resulting MFCC vector is obtained by retaining the first 12–20 coefficients, with coefficient c_0, the energy of the frame, often dropped from the feature vector. MFCCs were widely used in modelling with GMMs and HMMs, but their popularity has declined with the rise of deep learning methods.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi}{N}kn} \tag{5.1}$$

$$m = 1127 \ln(1 + \frac{f}{700}) \tag{5.2}$$

$$f = 700(e^{\frac{m}{1127}} - 1) \tag{5.3}$$

$$c_n = \sum_{m=1}^{M} \log \hat{y}_m \cos\left[\frac{\pi n}{M}\left(m - \frac{1}{2}\right)\right] \tag{5.4}$$

5.1.2 Spectrogram based features

The emergence of deep learning, particularly CNNs, has led to significant advances in computer vision tasks such as image classification surpassing traditional methods that relied on engineered features [93]. A similar paradigm shift can be observed in audio data analysis, where the audio signal is transformed into a two-dimensional TF-representation using the STFT and treated similarly to an image. This paradigm shift is reflected in Figure 5.1, showing the differences in extracting known features into a feature vector, and allowing the CNN to learn features directly from the spectrogram. Recent studies in both speech and bioacoustics tasks predominantly use the magnitude spectrogram as an input feature. However, creating a spectrogram involves several crucial hyperparameter choices [69], which can significantly impact the performance of a model. Parameters such as the window function type, frame length, and frame overlap require consideration. The frame length for example determines the trade-off between time and frequency resolution (higher time resolution at the expense of frequency resolution and vice versa), while the window function

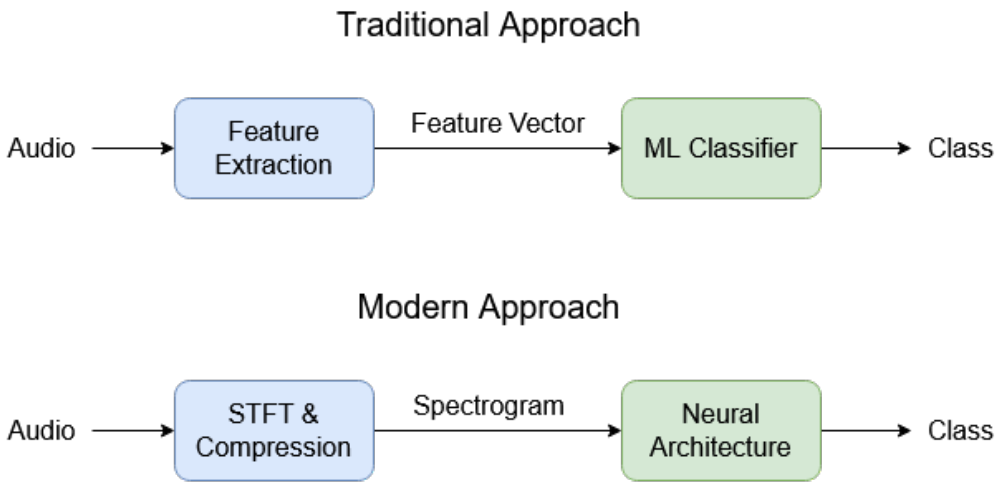

Figure 5.1: Traditional Feature extraction for speech and audio machine learning employed separate pipelines, extracting known features based on a set of predefined parameters. These features were then fed into the ML system. The modern approach is to transform the audio into a spectrogram representation, apply compression, and allow the convolutional layers of a CNN to extract features. Blocks in green are trainable.

used affects frequency selectivity, resolution, spectral leakage, and dynamic range. Although the magnitude spectrogram obtained from the STFT can be utilised as is, additional considerations for dimensionality reduction and dynamic compression will be discussed below.

Mel Spectrogram

The STFT generates a TF-representation that is uniformly sampled both in time and frequency. However, the uniform sampling in frequency may not be ideal as human perception of pitch is non-linearly related to frequency. To address this, a common approach is to convert the frequency axis of the spectrogram from a linear scale to the mel frequency scale, similar to the mel scale used in MFCCs as discussed earlier in Section 5.1.1.

The transformation from the linear spectrogram to the mel frequency spectrogram involves several steps. Starting with the STFT of the signal x[n] using a windowing function w[n] with Fast Fourier Transform (FFT) length N, the representation $\mathbf{X}$ is produced. The mel filterbank is defined by the matrix $\mathbf{\Psi_m}$, using M triangular filters, with centre frequencies linearly spaced on the mel scale between the minimum and maximum frequency range of the filterbank. The matrix $\mathbf{\Psi_m}$ is a linear transformation matrix projecting the FFT bins from

the STFT onto mel frequency bins. The resulting mel spectrogram matrix $\mathbf{M_x}$ is calculated by applying Equation 5.5, where the squared magnitude of $\mathbf{X}$ yields a linear spectrogram, and the matrix $\mathbf{\Psi_m}$ provides a frequency warping via linear transformation.

$$\mathbf{X} \in \mathbb{R}^{T \times N}$$

$$\mathbf{\Psi_m} \in \mathbb{R}^{N \times M}$$

$$\mathbf{M_x} = \mathbf{X}^{|\cdot|^2} \mathbf{\Psi_m}^{|\cdot|^2} \tag{5.5}$$

$$\mathbf{M_x} \in \mathbb{R}^{T \times M} \tag{5.6}$$

This results in a mel spectrogram characterised by a frequency axis that is logarithmic for frequencies above 1000 Hz and reduced in dimensionality ($T \times M$) compared to the original TF-representation ($T \times N$), where $M < N$. This reduction is beneficial in reducing the memory and computational requirements of network. Furthermore, shifts in harmonic signals appear to be linear shifts when scaled logarithmically, which may be a good match for CNNs, which detect linearly-shifted features reliably. When using the mel spectrogram as an input feature to a machine learning system, additional design choices include determining the number of filters in the filterbank and the desired frequency range. However, it is worth noting that the applicability of the mel scale in bioacoustics is debatable, as it is a perceptual scale based on human hearing and humans are rarely the intended recipients of such audio. An example of a mel spectrogram using 40 mel filters and logarithmic compression (see below) can be seen in Figure 5.2.

Spectrogram Dynamic Range Compression

A choice made in all TF-representations of audio is whether to compress the dynamic range of the representation. Compressing the spectral magnitudes can result in loudness equalisation, ensuring quieter events have similar prominence in the TF-representation as louder events. Dynamic range compression also helps in normalising the values before feeding them into a neural network, which is standard practice in machine learning. There are many

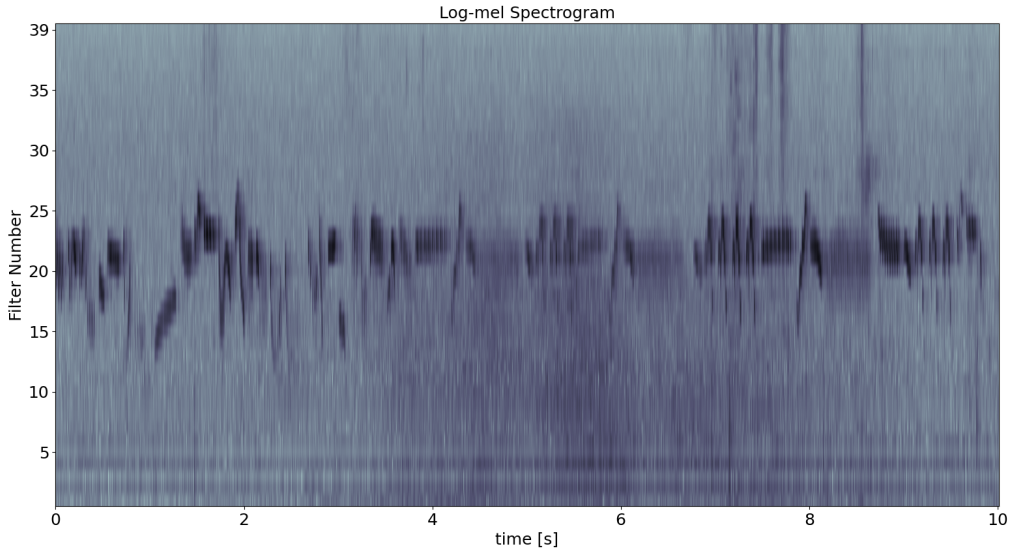

Figure 5.2: A mel spectrogram of audio containing birdsong from *BirdVox-DCASE-20k* [116]. The spectrogram energies are compressed logarithmically. The mel filterbank matrix is configured as 40 filters with a frequency range between 500 Hz – 16 kHz.

methods of static dynamic range compression:

- Logarithmic Compression
- Cubed Root Compression
- Tenth Root Compression

Cube root [72] and tenth root [165] compression involve taking the relevant root of each element in the TF-representation. These methods are designed to emulate the power laws of human hearing, replicating the non-linear relationship between sound intensity and perceived loudness. These root laws are more commonly used in analysis, and to enhance speech intelligibility to human listeners [119]. The most widely used compression method by far is logarithmic compression, which also aims to mimic the behaviour mentioned above.

However, these static compression methods have shortcomings:

1. The $\log$ function exhibits a singularity at 0, causing time-frequency bins with no energy to have infinite values after logarithmic transformation. Common approaches to

address this issue include the using the clipped logarithm ($\log(\max(\epsilon, x))$) or the stabilised logarithm ($\log(x + \epsilon)$). However, there is no consensus on whether one approach should be preferred.

2. The logarithmic and root transformations predominantly compress the dynamic range of low-level parts of the signal, such as segments containing near-silence. However, these low-level segments are typically the least informative part of the signal. There is also an added effect of accentuating the noise floor relative to the signal. This issue is less present when using tenth root compression [119], however in this case the overall dynamic range suffers.

3. The logarithmic and root transformations are dependent on the loudness of the signal. The values produced for two utterances will differ depending on how close the source is to the microphone. This can pose issues when far-field and near-field sources are both present in the recording.

5.1.3 Other Features

Constant-Q Transform

The Constant-Q Transform (CQT) [19] is related to, but different from, the DFT and can be used to create a TF-representation similar to a spectrogram. While the DFT samples frequencies uniformly, the CQT employs a logarithmic frequency sampling approach. The quality factor Q of a filter is defined as the ratio between the centre frequency and bandwidth of that filter ($Q = \frac{f_k}{\delta f_k}$). Unlike the DFT where Q decreases with increasing frequency, the CQT maintains a constant Q value at all frequencies. As a result, the CQT provides high frequency resolution and low time resolution at lower frequencies, while offering lower frequency resolution and high time resolution at higher frequencies. This characteristic mirrors the auditory system of humans and many other animals. Additionally, the CQT allows for centring at a desired frequency, and each octave consists of an equal number of filters. These properties make the CQT a commonly used feature in instrument recognition and related music tasks, and it has also found applications in bioacoustics. However, it is not as widely utilised as other techniques.

Sinusoidal Tracking

Sinusoidal tracking [81], sometimes referred to as frequency tracking, involves identifying sinusoidal components in a signal and estimating their amplitude, frequency and possibly phase. It encompasses the detection of fundamental frequency (F_0) as well as the tracking of harmonic content. Sinusoidal tracking has been employed in bird species detection as an alternative feature to MFCCs [79, 80]. It is typically used in conjunction with HMMs, similar to MFCCs, and has been found to be more noise robust than MFCCs when modelling bird vocalisations [79]. By employing HMMs, temporal modelling of the harmonic content and intensity of the signal becomes possible, enabling the identification of bird species from this modelling. Despite recent utilisation of sinusoidal tracking in certain studies on bird species identification [80], this feature remains less commonly employed compared to spectrogram-based features.

5.2 Learnable Frontends

In recent years, a number of promising frontends which learn from the data directly have been proposed. These *learnable frontends* turn what were previously *hyperparameters* in the non-learnable case, into *learnable parameters*. Some of the shortcomings of non-learnable frontends are the need to tune hyperparameters or utilise fixed filterbanks and static compression. These can be addressed by learnable frontends. This shift away from the contemporary method of extracting spectrograms, using fixed hyperparameters, from the audio and allowing the network to extract features, toward the approach of learnable frontends where all features are learned directly from the data, can be seen in Figure 5.3.

While the use of learnable frontends may incur additional computational costs, they can provide a representation of the input signal that is tailored to a specific task, architecture, and audio domain. In many cases, this representation is a TF-representation of the signal, similar to the one produced by the Short-Time Fourier Transform (STFT). Other frontends do not provide TF-representations of the audio signal, but instead learn interpretable spectro-temporal features [157].

Learnable frontends which provide TF-representations encompass a wide range of

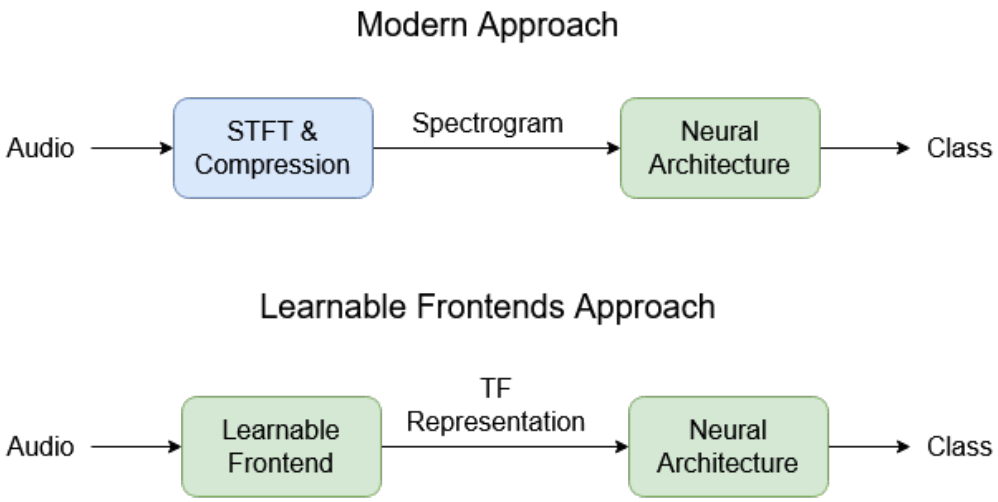

Figure 5.3: Contrasted to the modern approach of feature extraction, learnable frontends usually aim to compute or modify a Time-Frequency representation in a data driven manner, which is optimised alongside the network. Blocks in green are trainable.

functionality. These functions include filterbanks [24, 161, 216, 217], temporal downsampling [216, 217] and magnitude compression [166, 211]. Some frontends specialise in one particular aspect, whereas others combine these elements in a pipeline (such as LEAF [216]). Many learnable frontends incorporate learnable filterbanks. The authors in [166] categorise frontends utilising learnable filterbanks based on two criteria: (1) the domain of operation, i.e. whether processing takes place in the time or frequency domain; and (2) whether filter responses are learned directly or via a parameterised function.

1. The domain of operation has a large impact on the type of filtering that can be carried out. Certain types of filters are more easily implemented in the frequency domain, such as rectangular and triangular filters. Furthermore, time-domain filtering requires more operations (achieved via convolution and on the order of $\mathcal{O}(n^2)$) than filtering in the frequency-domain using overlap-add methods [181] (on the order of $\mathcal{O}(n \log n)$ [36]). Frequency-domain filtering in the context of CNNs requires computation of the STFT instead of the FFT. Usage of the STFT requires additional design choices, such as windowing function and frame/overlap settings. Deep learning applications benefit from the use of GPUs and other specialised hardware, making convolution more practical due to increased parallelism. Convolutional layers in CNNs efficiently

implement a cross-correlation operation (this mixup of nomenclature does not effect results in practice), which is equivalent to a convolutional operation if the time-domain response of a filter is even ($h[n] = h[-n]$) or the time-domain response is pre-reversed. While there are many examples of learnable filterbanks implemented in the frequency domain, such as those in [24, 35, 58, 164], time-domain filterbanks have become more prevalent in recent years due to their easy implementation within CNNs, as seen in [137, 151, 216, 217].

2. The direct learning of filter coefficients, as demonstrated in [217], offers significant flexibility in terms of filter properties and has the potential to improve performance [3]. However, this freedom comes at the cost of increased training time and an increase in the number of model parameters. Additionally, while the learned filters may be highly suitable for the audio data and task, analysing the resulting filter properties or frequency response may not be straightforward. Although this does not affect raw performance, conducting further analysis on the filters can provide valuable insights about the data and what aspects of it the neural network deems important for classification.

   On the other hand, coefficients generated through a parameterised function [151, 216] not only reduce the number of parameters but these parameters have an interpretable meaning with regards to the filter (typically centre frequency and bandwidth). These parametric functions can be implemented in either the time or frequency domain and have been extensively studied. Although there might be a slight degradation in performance, having fewer parameters that carry direct meaning can be beneficial, as it paves the way to explainability in the system.

Regardless of form, learnable filterbanks require an intialisation strategy to set the initial values of the filters. Typically, the filters are initialised based on a static filterbank, with the mel scale [135, 180] being a common choice, especially for tasks involving human speech. In Section 5.4, we discuss other initialisation strategies which may be utilised.

What follows in this section is an overview of four learnable frontends, which are implemented as layers within a neural network. These learnable frontends include

Spectro-Temporal Filters (STRF) [157], Time-Domain Filter Banks (TD) [217], Per-Channel Energy Normalisation (PCEN) [211] and Learnable Audio Frontend (LEAF) [216]. A technical explanation of each frontend is provided, and in Section 5.3 each frontend described is evaluated on a shared bioacoustics task. Detailed information regarding this experiment can be found in Section 5.3, with results and discussion in Section 5.3.4.

5.2.1 Spectro-Temporal Filters

Spectro-Temporal Filters (STRF) were proposed by Riad et al. [157] in 2021 as a general purpose acoustic frontend. STRF consists of a set of learnable two-dimensional Gabor filters designed to detect spectro-temporal modulations within a spectrogram. A primary objective of the work was to guide neural networks in a more biologically inspired direction by imposing limitations on the extracted features from a TF-representation. To achieve this, the filters are constrained to the space of Gabor functions, denoted as $g(t, f)$, as defined by Equation 5.7.

These Gabor filters are composed of a sinusoidal plane wave $s(t, f)$ as defined in Equation 5.8, and modulated by a Gaussian function $w(t, f)$ as defined by Equation 5.10. The sinusoidal plane wave $s(t, f)$ is parameterised by F, which determines the frequency of the wave and R_γ (Equation 5.9), which controls the orientation of the sinusoid in two-dimensional space. The Gaussian window $w(t, f)$ is always centred in the middle of the filter and is solely governed by the standard deviations of the Gaussian in the time dimension σ_t and frequency dimension σ_f. Consequently, each filter (denoted by k), is characterised by four values represented as $F_k, \gamma_k, \sigma_{f_k}, \sigma_{t_k}$.

$$g_k(t, f) = s_k(t, f) \cdot w_k(t, f) \tag{5.7}$$

$$s_k(t, f) = e^{j(2\pi(F_k R_{\gamma_k}))} \tag{5.8}$$

$$\text{where,} \quad R_{\gamma_k} = t\cos(\gamma_k) + f\sin(\gamma_k) \tag{5.9}$$

$$w_k(t, f) = \frac{1}{2\pi\sigma_{t_k}\sigma_{f_k}} e^{-(\frac{1}{2}(t^2/\sigma_{t_k}^2 + f^2/\sigma_{f_k}^2))} \tag{5.10}$$

As previously mentioned, the input to the STRF is an existing TF-representation denoted by

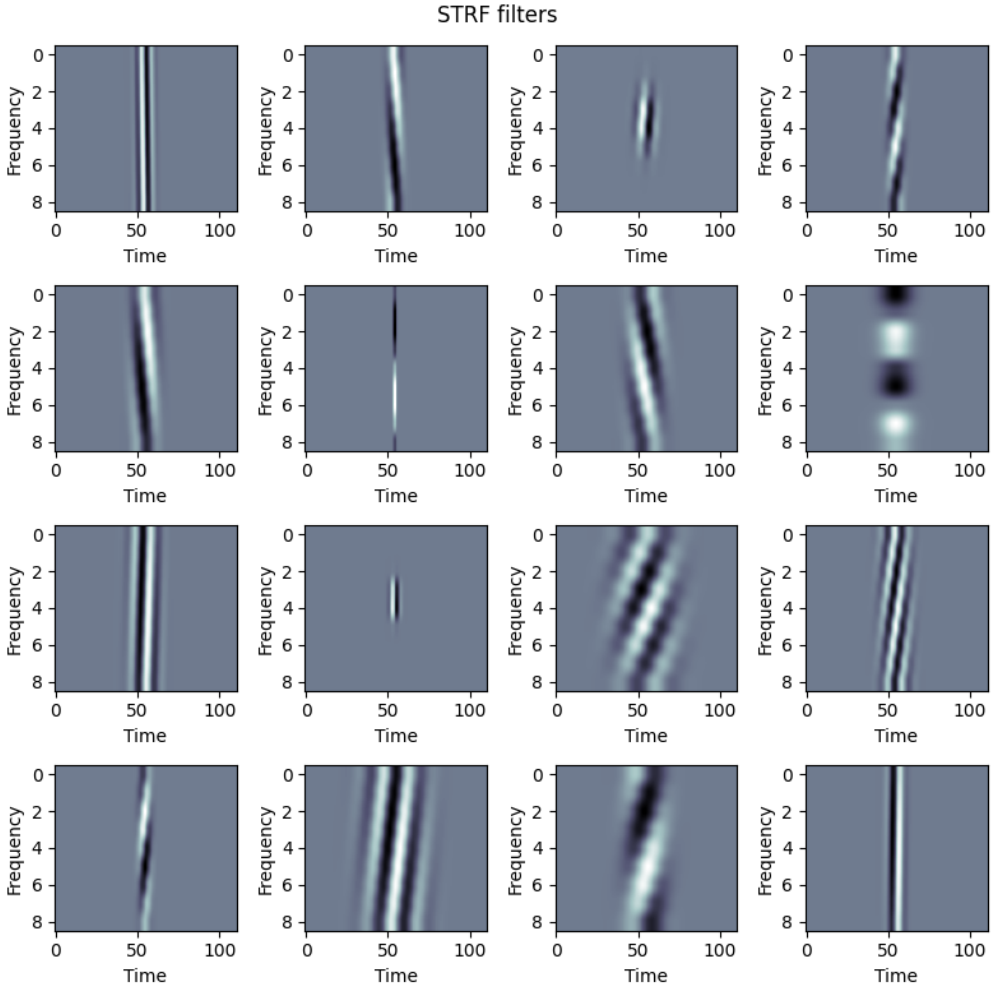

Figure 5.4: Examples of learned STRF filter kernels learned as part of the bird activity detection task outlined in Section 5.3. This figure shows the weights of 16 randomly selected filters. As in [157], each filter is provided with a receptive field of 9 frequency channels in the frequency domain and 1.1s in the time domain.

$E(t, f)$. Unlike the other frontends detailed in this section, STRF does not construct or modify a TF-representation. Instead, it limits the filter kernels to yield a more interpretable and biologically inspired set of features. Examples of these learned filter kernels can be found in Figure 5.4, which presents the weights of 16 out of 64 learned filters. Each filter is provided with a receptive field of 9 frequency channels in the frequency domain and 1.1s in the time domain. This characteristic sets it apart from the other evaluated frontends in this study. For this particular study and in similar works, the number of Gabor filters used is 64. The resulting features are represented as $\boldsymbol{Z}$ and defined by Equation 5.11.

$$\mathbf{Z}(t, f, k) = \sum_{u,v} E(u, v) g_k(t - u, f - v) \tag{5.11}$$

The parameters associated with each learned STRF filter $F_k, \gamma_k, \sigma_{f_k}, \sigma_{t_k}$ have direct relationships to amplitude modulation ω_k and frequency modulation Ω_k where $\omega_k = F_k \cos(\gamma_k)$ and $\Omega_k = F_k \sin(\gamma_k)$. This observation highlights that STRF is designed to capture patterns in spectral and temporal modulation, as it encourages the learning of such characteristics.

5.2.2 Time-Domain Filter Banks

Time-Domain Filter Banks (TD) were introduced by Zeghidour et al. [217] in 2018 as a learnable alternative to the widely used mel spectrogram. In their work, the authors propose a collection of learnable filterbanks which are first initialised to a known scale and subsequently fine-tuned in conjunction with the rest of the model. This fine-tuning process aims to optimise the filters specifically for a given application, task, or audio type. The TD approach builds upon previous work by Andén and Mallat [1], who presented an approximation of the mel frequency spectrum in the time domain. The frequency-domain calculation of the mel spectrogram has been discussed above in Section 5.1.2.

Consider the mel spectrogram as a continuous representation in both time and frequency domains, Equation 5.13 describes the averaging of spectrogram energy $\hat{x}_\phi(t, \omega)$

(Equation 5.12) with mel-scale filters $\hat{\psi}_f$ where f is the centre frequency of each filter $\hat{\psi}_f(\omega)$. The window function $\phi(t)$ has a length of T, and Q denotes the quality of the filter.

$$\hat{x}_\phi(t,\omega) = \int x(\tau)\phi(\tau - t)e^{-j\omega\tau}\, d\tau \tag{5.12}$$

$$M_x(t,f) = \frac{1}{2\pi}\int |\hat{x_\phi}(t,\omega)|^2 |\hat{\psi}_f(\omega)|^2\, d\omega \tag{5.13}$$

$$= \int |x_\phi * \psi_f|^2(v)\, dv \text{ (Parseval–Plancherel identity)} \tag{5.14}$$

$$= \int \left| \int x(u)\phi(u-t)\psi_f(v-u)\, du \right|^2 dv \tag{5.15}$$

If, $f \gg QT^{-1}$ then

$$M_x(t,f) \approx \int \left| \int x(u)\psi_f(v-u)\, du \right|^2 |\phi(v-t)|^2\, dv \tag{5.16}$$

$$= |x * \psi_f|^2 * |\phi|^2(t) \tag{5.17}$$

This time-domain approximation is the basis for the TD frontend. the formulation of TD (Equation 5.18) is identical to the time-domain approximation in Equation 5.17. In TD, the window function $\phi(t)$ is specified as a Hanning window and can be fixed, however the coefficients for this windowing function can also be learned alongside the filterbank. The number of filters can be adjusted as a hyperparameter, with the original implementation using $N = 40$ filters spanning 64 Hz to 8 kHz. Each filter ψ_n is initialised with a Gabor wavelet (as described in Equation 5.19), which is parameterised by the desired centre frequency η_n of the respective filter and a parameter inversely proportional to the desired bandwidth σ_n.

$$\text{TD}(t,n) = |x * \psi_n|^2 * |\phi|^2(t) \tag{5.18}$$

$$\psi_n(t) = e^{2\pi j \eta_n t} \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{t^2}{2\sigma_n^2}} \tag{5.19}$$

Although initialised by the Gabor wavelet specified above, TD learns the impulse responses of each filter directly. The learnable parameters in TD are not centre frequency and bandwidth,

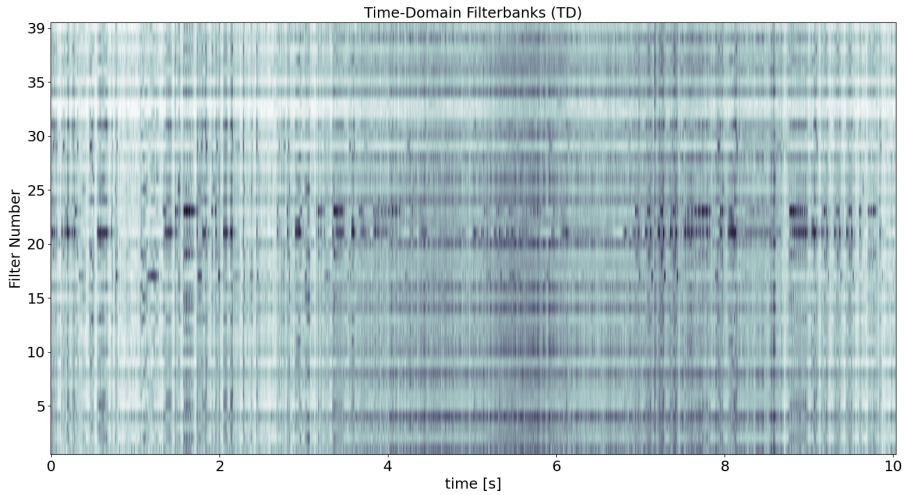

Figure 5.5: TF-representation generated by TD, after training, of audio containing birdsong from *BirdVox-DCASE-20k* [116]. TD applies logarithmic compression of magnitude by default. Note the absence of specific frequencies on the y-axis, as the learned filters do not have centre frequencies and are not well ordered.

but each sample or coefficient in the filter impulse response ψ_f. While this approach can result in a well-performing filterbank that is highly adapted to a specific task, it entails increased computational costs and a larger memory footprint. Moreover, the resulting filters may also be more difficult to analyse post hoc. An example of TDs output after training on a birdsong task can be seen in Figure 5.5.

TD offers additional features such as options for learnable pre-emphasis of the signal and a learnable windowing function (mentioned above). However, the original implementation paper [217] reports that these features did not offer reliable increases in performance. The authors also note that the learned window functions were nearly identical to their initialisation, and many of the learned filters exhibited similarities to the initial filters. TD does not include learnable compression. In its default configuration a logarithmic compression is used. It is worth noting that TD forms the background for the design of another frontend evaluated in this study, LEAF.

5.2.3 Per-Channel Energy Normalisation

Per-Channel Energy Normalisation (PCEN) was introduced by Wang et al. [211] in 2017 as a solution addressing challenges in far-field keyword spotting. Noting how the usage of DNNs had improved the performance of automatic speech recognition, robustness to far-field recordings and noise were still an issue. Keyword spotting, a task similar to automatic speech recognition, often involves far-field audio as input.

The authors identify that the log-mel spectrogram is perhaps the most commonly used input feature in DNN-based acoustic modelling. However, they also note three shortcomings associated with using the log-mel spectrogram, particularly in regard to using logarithmic dynamic range reduction in the spectrogram. These shortcomings, which have been discussed in Section 5.1.2, are summarised here for convenience.

1. The $\log$ function exhibits a singularity at 0, causing time-frequency bins with no energy to have infinite values after logarithmic transformation. Common approaches to address this issue include the using the clipped logarithm ($\log(\max(\epsilon, x))$) or the stabilised logarithm ($\log(x + \epsilon)$), but the choice between these seems arbitrary and ad-hoc.

2. The logarithmic transformation predominantly compresses the dynamic range of low-level parts of the signal, such as segments containing near-silence. However, these low-level segments are typically the least informative part of the signal. There is also an added effect of accentuating the noise floor relative to the signal.

3. The logarithmic transformation is dependent on the loudness of the signal. while it reduces the dynamic range, it does no provide a means to compress near-field and far-field sources to equal loudness.

PCEN is introduced as a viable alternative to log compression, offering the advantage that it can be optimised alongside the model. PCEN comprises two key operations, namely AGC and DRC, both of which are learned per frequency channel. A detailed asymptotic analysis of PCEN is provided by Lostanlen et al. in [115], while this thesis provides a brief overview of PCEN's operation.

$$M(t,f) = (1-s)M(t-1,f) + sE(t,f) \tag{5.20}$$

$$G(t,f) = \frac{E(t,f)}{(M(t,f)+\epsilon)^{\alpha}} \tag{5.21}$$

The input to PCEN is a TF-representation, often an uncompressed mel spectrogram, although other spectrogram-like TF-representations are also valid input [216]. This input TF-representation is denoted as $E(t,f)$, an example can be found in Figure 5.6(B). The smoothed spectrogram $M(t,f)$ (defined in Equation 5.20, smoothing is implemented through an IIR filter), serves as an estimate of the background noise and is learned per frequency band. An example of a smoothed spectrogram $M(t,f)$ can be seen in Figure 5.6(C). Equation 5.20 is effective for stationary noise sources, any audio which exhibits a temporal or frequency modulation lower than the cutoff frequency of the smoother will be subject to gain reduction in the AGC step. The AGC stage is described by Equation 5.21 and involves dividing the input TF-representation $E(t,f)$ by the smoothed TF-representation $M(t,f)$, thereby highlighting changes relative to the recent spectral history along the temporal axis [115]. Example output of this AGC operation can be found in Figure 5.6(D). The AGC operation is controlled by two parameters, s and α, both of which can be learned on a per-channel basis. In Figure 5.6, these parameters are set to $\mathbf{s} = 0.05$ and $\alpha = 0.98$. It should be noted that ϵ could also be parameterised, although in most scenarios it is typically set as a hyperparameter as its primary purpose is to prevent division by 0. Typically, ϵ is set to 10^{-6}, representing an arbitrarily small number.

$$\text{PCEN}(t,f) = (G(t,f)+\delta)^{r} - \delta^{r} \tag{5.22}$$

$$\Rightarrow \text{PCEN}(t,f) = \left(\frac{E(t,f)}{(M(t,f)+\epsilon)^{\alpha}} + \delta\right)^{r} - \delta^{r} \tag{5.23}$$

The second operation of PCEN involves DRC, which is achieved by the addition of a positive offset δ to $G(t,f)$ and can be seen in Equation 5.22. The result is then an element-wise

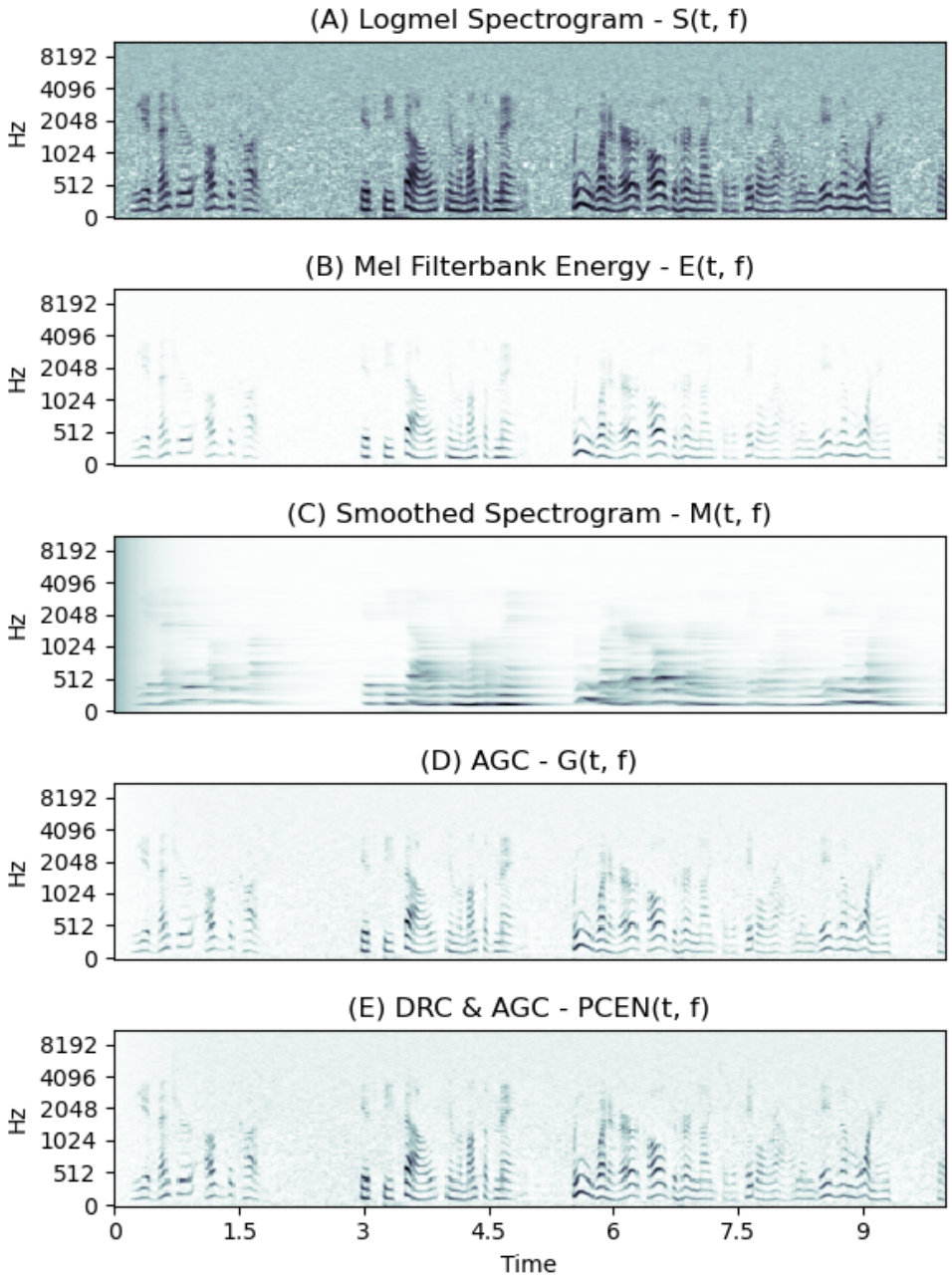

Figure 5.6: (A) shows a log-mel spectrogram of human speech ('libri2' example from Librosa) with additive white noise. (B) shows the uncompressed mel spectrogram energy. (C) shows a smoothed version of (B), based on Equation 5.20, $\mathbf{s} = 0.05$. (D) is the output of the AGC operation of PCEN defined by Equation 5.21, with $\alpha = 0.98$. (E) is the result of the DRC defined by Equation 5.22, $\delta = 2$ and $\mathbf{r} = 0.5$, and the final PCEN output (Equation 5.23).

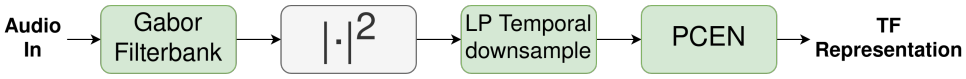

Figure 5.7: Signal flow from input audio to final Time-Frequency representation in LEAF. Blocks in green are trainable.

exponential operation, followed by removal of the bias parameter. The parameter δ corresponds to the threshold parameter in dynamic compression, while the parameter r corresponds to the compression ratio after this threshold. Higher values of r correspond to less compression and a more linear transfer function. This is also the final output of PCEN, and an example of this output can be seen in Figure 5.6(E), with DRC parameters of $\mathbf{r} = 0.5$ and $\delta = 2$.

The complete formulation of PCEN can now be seen in Equation 5.23 and is based on the set of parameters $\mathbf{s}, \alpha, \delta, \mathbf{r}$ where each parameter is learned per-channel. The output of PCEN is a TF-representation of the signal, preserving the same dimensions as the input TF-representation. PCEN has demonstrated its effectiveness in various tasks, including keyword spotting [6, 169] and bioacoustics [37, 42, 114, 117].

5.2.4 (Efficient) Learnable Audio Frontend

Learnable Audio Frontend (LEAF) was proposed by Zeghidour et al. [216] in 2021 as a learnable frontend that incorporates both learnable filterbanks and learnable compression. It combines elements from TD (learnable filterbanks and learnable low-pass filtering) and PCEN (the entire PCEN compression layer). Therefore LEAF consists of three learnable layers, and one fixed operation. The signal flow in LEAF is shown in Figure 5.7.

The operation of LEAF can be described as follows. The input audio signal x is convolved with a set of band-pass filters ψ (described in Equation 5.24) of length W ($|t| \leq \frac{W}{2}$) in the time-domain. These filters take the form of Gabor wavelets and are parameterised by their centre frequency ($\eta_n \in [0, 1]$), and the standard deviation of the Gaussian envelope in the time domain ($\sigma_{n_{bw}} \in (0, \frac{F_s}{W+1})$). Notably, the standard deviation of the Gabor wavelet in the time domain (i.e. the width of the filter kernel), is inversely proportional to the bandwidth of

the filter in the frequency domain. This relationship holds true due to the Fourier transform of a Gaussian function also being a Gaussian function. The output of the band-pass filtering operations ($\mathbf{x}_\psi$) yields time sequences with the same temporal resolution as the input signal x. To ensure that the output of the filterbank is analytic (i.e. the frequency responses contains no negative frequency components), the squared modulus of these time sequences are calculated (as indicated in Equation 5.25 for both the convolution and squared modulus operations).

$$\psi_n(t) = e^{2\pi j \eta_n t} \frac{1}{\sqrt{2\pi}\sigma_{n_{bw}}} e^{-\frac{t^2}{2\sigma^2_{n_{bw}}}} \tag{5.24}$$

$$x_{\psi_n}(t) = |x * \psi_n|^2(t) \tag{5.25}$$

In order to create an uncompressed TF-representation of the signal, the time sequences must be downsampled. LEAF applies a learnable low-pass filter (one per frequency band) to the time sequences $\mathbf{x}_\psi$. These learnable low-pass filters are Gaussian filters (described in Equation 5.26) and have a similar form to the filters used in TD, as can be observed by comparing Equation 5.18 and Equation 5.27. This is followed by sub-sampling via strided convolution to generate a TF-representation. The final component of LEAF is a fully learnable PCEN layer which is defined above by Equation 5.23 and its operation has been explained in Section 5.2.3. Equation 5.28 represents the resulting TF-representation, which serves as the output of 'Learnable Audio Frontend'. An example of this output, with LEAF having been trained on a bird audio task, can be seen in Figure 5.8.

$$\phi_n(t) = \frac{1}{\sqrt{2\pi}\sigma_{n_{lp}}} e^{-\frac{t^2}{2\sigma^2_{n_{lp}}}} \tag{5.26}$$

$$\begin{aligned} \hat{X}(t,n) &= x_{\psi_n} * |\Phi_n|^2(t) \\ &= |x * \psi_n|^2 * |\phi_n|^2(t) \end{aligned} \tag{5.27}$$

$$\mathsf{LEAF}(t,n) = \mathsf{PCEN}(\hat{X}(t,n)) \tag{5.28}$$

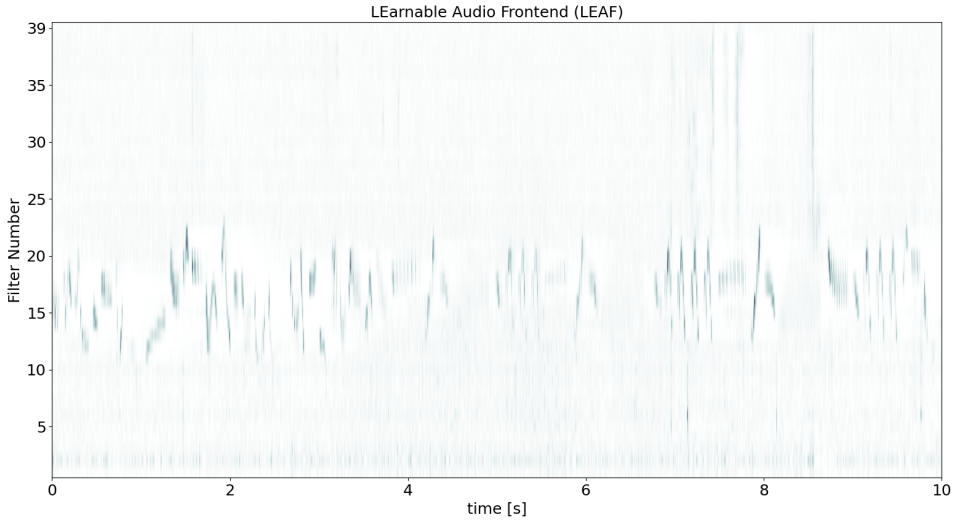

Figure 5.8: TF-representation generated by LEAF, after training, of audio containing birdsong from *BirdVox-DCASE-20k* [116]. LEAF includes a trainable PCEN layer, which is the method of dynamic range compression utilised in this TF-representation. Note the absence of specific frequencies on the y-axis, as the learned filters are not well ordered.

An optimised implementation of LEAF named Efficient Learnable Audio Frontend (eLEAF) has been developed by Schlüter and Gutenbrunner [166] in 2022. The primary goal of eLEAF is to increase throughput during training (this includes both the forward and backward passes) and inference when compared with the original LEAF implementation. This is accomplished through two key modifications to LEAFs internal workings:

1. Dynamic sizing of convolution window sizes and stride lengths for subsets (or 'groups') of filters. By adjusting the size of the filters, the computational complexity of the convolution operations within the network is reduced. As the bandwidth in the frequency domain of a filter increases, the width of the filter kernel in the time domain decreases, concentrating filter energy in fewer coefficients. To optimise the filterbank, filter length is truncated based on the calculated value $L_f = b\sigma_{n_{bw}}$, where b is a hyperparameter. A similar process is applied to determine stride lengths $L_s = d\pi/v$, where v is the centre frequency of the filter and d is a hyperparameter. This technique can be further improved by applying it to g groups of subsequent filters, with the largest filter length and smallest stride to be used for the whole group. Through a

hyperparameter grid-search, the authors in [166] found that values of $b = 6$, $d = 16$ and $g = 8$ provide improved efficiency while maintaining good performance.

2. Replacement of the PCEN layer with a more easily implemented layer. The PCEN layer requires the calculation of an exponential moving average along the temporal dimension, which is not well-suited to massive parallelism of a GPU. The authors in [166] replicate some of the effects of PCEN by incorporating a per-channel learnable logarithmic compression, subtraction of the median along the temporal axis, and batch normalisation.

By implementing these modifications, eLEAF achieves a substantial $37x$ increase in throughput compared to classic LEAF. However, in this work, when eLEAF is employed a PCEN compression layer is used instead of the modified compression layer. The modifications to the filterbank layer contribute more to eLEAF's increase in throughput than the modified compression layer. Furthermore, PCEN is much more widely understood and analysed [115] compression layer which has seen widespread use.

5.3 Evaluating Learnable Acoustic Frontends on a Bird Activity Detection Task

The learnable frontends discussed previously have been extensively evaluated on various speech datasets and tasks [6, 157, 169, 216, 217]. Some of them have even been evaluated on a bioacoustics task such as bird species identification. However, prior to work by the author of this thesis in [3], there was no dedicated comparative assessment of these methods specifically for bird activity detection. The study in [3] aimed to compare traditional (i.e. non-learnable) with learnable frontends using the same datasets, task and model architecture. Evaluating existing approaches and their applicability to bird audio was crucial, as an increasing number of deep learning systems are incorporating 'off-the-shelf' learnable frontends, presuming that the features learned from the data are optimal.

To evaluate each frontend, independent supervised models were trained on a Bird Activity Detection (BAD) task. The BAD task has been described elsewhere in Section 2.2.2, but is

restated here for ease of reading. BAD serves as a crucial initial step for any subsequent population analysis and crucial for assessing the suitability of these learnable frontends in bird bioacoustics research. In this specific case, the activity detection task involves binary classification, determining whether a 10-second clip of audio contains bird vocalisations or not. It is important to note that BAD is a species agnostic task, meaning the system is expected to generalise to different species and vocalisation types not encountered during training.

The performance of the four described learnable frontends, along with three non-learnable spectrogram baselines, was measured and compared in terms of their ability to detect species-agnostic bird activity. The evaluation aimed to provide insights into the effectiveness and suitability of these frontends for bird activity detection tasks.

The list below presents the non-learnable and learnable frontends used in this evaluation.

- **Non-learnable Frontends**
  - Linear Spectrogram (spect)
  - Mel Spectrogram (mel)
  - Log-Mel spectrogram (logmel)
- **Learnable Frontends**
  - Spectro-Temporal Filters (STRF)
  - Time-Domain Filter Banks (TD)
  - Per-Channel Energy Normalisation (PCEN)
  - Learnable Audio Frontend (LEAF)

The datasets involved in testing contain a large variety of bird species and call types, as well as many noise sources (e.g. wind, traffic, human activity and speech). In the following sections the datasets used will be described in detail, followed by a brief description of the backend model used in classification. Evaluation and testing methods are then described, followed by the presentation and discussion of the results of this experiment. The study

Dataset	Pos. (Bird Present)	Neg. (Bird Absent)	Total
BirdVox-DCASE-20k	10017	9983	20000
freefield1010	5755	1935	7690
warblrb10k	6045	1955	8000
Totals	21817	13873	35690

Table 5.1: Details of datasets included from the DCASE2018 Bird Audio Detection Challenge. Positive in this context are recordings with a bird present. Negative are recordings where no bird is present.

described in this section was originally published as [3].

5.3.1 Dataset

The datasets used in this study were obtained from the DCASE2018 Bird Audio Detection Challenge [187]. The challenge organisers provided annotated datasets from three different sources, as described in Table 5.1. The inclusion of multiple datasets aimed to ensure a degree of generality.

One dataset, *BirdVox-DCASE-20k* [116], consists of passively recorded data collected from remote monitoring projects. The other two datasets, *freefield1010* [185] and *warblrb10k*[1], are actively recorded and crowdsourced recordings contributed by the freesound project and users of the Warblr bird recognition app, respectively.

In total, the three datasets comprise 35690 recordings. Each recording is 10 seconds in length and is sampled at 44.1 kHz. The audio data has been normalised to -2 dBFS. The datasets provide clip-level annotations indicating the presence (positive label) or absence (negative label) of bird activity. Bird activity could be localised to a few milliseconds or span the entire clip. The class distribution is unbalanced, with a ratio of 60% positive labels to 40% negative labels. No class balancing was performed prior to training.

Among the datasets, *BirdVox-DCASE-20k* is considered more challenging due to the high presence of environmental noise and far-field recordings. The *freefield1010* and *warblrb10k* contain near-field recordings but still contain large amounts of human-generated noise. For the training, validation, and test datasets, a split of 70% for training, 15% for validation, and

[1]Accessible via: https://archive.org/details/warblrb10k_public

15% for testing was used. When splitting the datasets, the relative proportions and class balance of each dataset were maintained.

5.3.2 Model

The network architecture for all experiments in this section is EfficientNet-B0 [190], with one head node providing classification output. The most popular model architectures for bioacoustics work are ResNet or VGG-based [184]. However, EfficientNet based models offer a good compromise between computational resources and accuracy, and can be deployed on edge-based systems. Pre-trained weights are not used to initialise the network. Each model is trained from scratch using Binary Cross Entropy loss and the ADAM optimiser [90] with an initial learning rate of 10^{-3}. The learning rate is adaptive, with a 10x reduction when the validation loss reaches a plateau.

The relevant hyperparameters and initial settings for each frontend are detailed in Table 5.2. We utilise a window length of 10 ms in all our FFTs with an overlap of 75% (i.e. a hop length of 2.5 ms). This allows for increased time resolution at the expense of frequency resolution, which is will be reduced by the introduction filterbanks in most of the tested frontends. STRF utilises 64 filters, this is the default number of filters used by Riad et al. [157] in their original implementation. The number of filters used in all filterbank-based frontends is 40, the default number of filters used in LEAF. Each filterbank spans between 500 Hz and 16 kHz, with centre frequencies linearly spaced along the mel scale. This aligns with the typical frequency range of bird vocalisations [63, 158]. PCEN parameters are initialised based on defaults from Lostanlen et al. [115]. While the mel/logmel filterbanks remain fixed, TD and LEAF are only initialised to these values and are permitted to optimise over the course of training.

5.3.3 Evaluation and Testing

To evaluate the performance of each frontend, the following analysis is conducted. First, the accuracy of the model using a particular frontend is reported when evaluated on the entire test set. Accuracy is further broken down by each individual dataset. Accuracy is reported as both the positive and negative class are of equal importance in this task, this is in spite of

spect	N_{fft}	128
	Window Length	10 ms
	Hop Length	2.5 ms
	Window Function	Hamming Window
mel/logmel	N_{fft}	128
	Window Length	10 ms
	Hop Length	2.5 ms
	Window Function	Hamming Window
	No. Filters	40
	Min Freq.	500 Hz
	Max. Freq	16000 Hz
STRF	No. Filters	64
TD	Window Length	10 ms
	Hop Length	2.5 ms
	Filterbank Init.	Mel
	No. Filters	40
	Min Freq.	500 Hz
	Max. Freq	16000 Hz
PCEN	$\boldsymbol{s}$	0.025
	Init. $\boldsymbol{\alpha}$	0.8
	Init. $\boldsymbol{\delta}$	10.0
	Init. $\boldsymbol{r}$	4.0
LEAF	Window Length	10 ms
	Hop Length	2.5 ms
	Filterbank Init.	Mel
	No. Filters	40
	Min Freq.	500 Hz
	Max. Freq	16000 Hz
	PCEN Settings	As above

Table 5.2: Hyperparameters and initial settings of each frontend.

the class imbalance.

To assess the statistical significance of the results, 30 random independently distributed subsets created with replacement of the test set are created and their accuracy evaluated per model. One-way Analysis of Variance (ANOVA) and Tukey's Honestly Significant Difference test (HSD) [199] are employed to determine if there are significant differences in model performance (and consequently, frontend performance). These statistical tests rely on three assumptions:

1. The observations (accuracy per subset) are independent.
2. Homogeneity of variances between groups (the largest sample variance in a group is less than twice the variance of the smallest sample variance in a group).
3. The distribution of the residuals are normal (i.e. the observations are normally distributed).

Assumption (1) is satisfied due to the sampling strategy used to create the subsets of test data. Assumption (2) is easily evaluated using the sample means and variances from each group, and is found to be satisfied. Assumption (3) can be tested using a normality test, such as the Shapiro-Wilk [170] test.

The Shapiro-Wilk test assesses the null hypothesis that a collection of samples $\mathbf{x}$ is drawn from a normal distribution. If the p-value is less than 0.05, the null hypothesis is rejected, indicating that the samples were not drawn from a normal distribution. The number of samples is equal to the number of independently distributed subsets created (i.e. $N = 30$). The normality tests indicate that all data are distributed normally, which allows for the use of one-way ANOVA testing.

One-way ANOVA testing is employed to determine if the mean accuracy of at least one of the models significantly differs from the others. Verifying this allows for a post-hoc pairwise comparison to identify which models have significantly different results from each other. Tukey's HSD is used to identify pairs of distributions which differ significantly from each other. This analysis determines whether the test set results' distributions significantly differ between the different frontends. If there is no significant difference between two frontends on

the whole test set, then further analysis will be carried out by comparing their accuracy per dataset.

It is important to note that if there is no significant difference between two frontends, this result may only apply to this specific task. There could be other considerations beyond overall performance that need to be taken into account.

These statistical tests are also employed in Section 5.4, but are less robust compared to training and evaluating multiple models and assessing statistical significance based on their collective performance. Employing multiple models captures variability in training, optimisation, and batch composition, providing a better representation of performance and whether the results are reproducible. Chapter 6 utilises this approach after discussions with fellow researchers highlighted some concerns about variability in training.

5.3.4 Results & Discussion

The overall results are presented in Table 5.3 and Figure 5.9. Table 5.3 reports the performance of each frontend on the test data, allowing for an overall system ranking. From this we can see that PCEN is the best overall performer with an accuracy on the test set of 89.9%. Using the same dataset (but not the same training, validation or testing splits), Liaqat et al. [104] attain an accuracy of 89.0% by employing a system based on 'bulbul' [67]. They employ a log-mel spectrogram with 160 filters, resulting in a significantly larger feature with finer frequency resolution compared to the 40 filters used here, in addition to extra audio preprocessing and domain tuning. Figure 5.9 details performance per dataset, providing insight into how each frontend performs under the differing conditions of each dataset.

Table 5.4 presents the results of the pair-wise significance testing. It can be observed that the majority of the pair-wise testing results are statistically significant. However, some results are only statistically significant when considering the individual datasets within the test set separately. While the mean values for the entire test set may not differ significantly, the mean values for specific datasets considered individually do show significant differences. These particular comparisons are marked with a different square symbol ($\square$) to differentiate them. Most of these comparisons involve frontends that use a mel spectrogram (either Mel

Frontend	**Test Set Accuracy (%)**
spect	78.4
mel	71.7
logmel	70.4
STRF	71.3
TD	87.6
PCEN	**89.9**
LEAF	83.7

Table 5.3: Accuracy of EfficientNet-B0, broken down by frontend, on the full test dataset. The best result is marked in **bold**.

	spect	mel	logmel	STRF	TD	PCEN
spect	N/A					
mel	■	N/A				
logmel	■	□	N/A			
STRF	■	□		N/A		
TD	■	■	■	■	N/A	
PCEN	■	■	■	■	■	N/A
LEAF	■	■	■	■	■	■

Table 5.4: Significance tests on pairwise-comparisons using Tukey's HSD. Cells marked with ■ indicate statistically significant results ($p < 0.05$) on the entire test set. Cells marked with □ indicate statistically significant results on at least one dataset.

Spectrogram or Log-Mel spectrogram). Interestingly, the pairwise comparison of STRF and logmel also exhibits this behaviour. The underlying reasons for the STRF–logmel pairwise comparison's lack of statistical significance are likely similar to the reasons behind why the STRF–mel comparison is not statistically significant.

There is no significant difference in performance between the STRF and logmel results across all datasets, indicating comparable performance. This can be attributed to the fact that STRF operates differently and serves a different purpose compared to the other learnable frontends tested. Instead of producing or enhancing a TF representation, STRF imposes biologically inspired constraints on the interpretation of the input TF-representation. The features learned from STRF are then input into the larger EfficientNet-B0 backend. By default, the input to STRF is a Log-Mel spectrogram. Although Log-Mel spectrogram + STRF + EfficientNet does not outperform Log-Mel spectrogram + EfficientNet, the

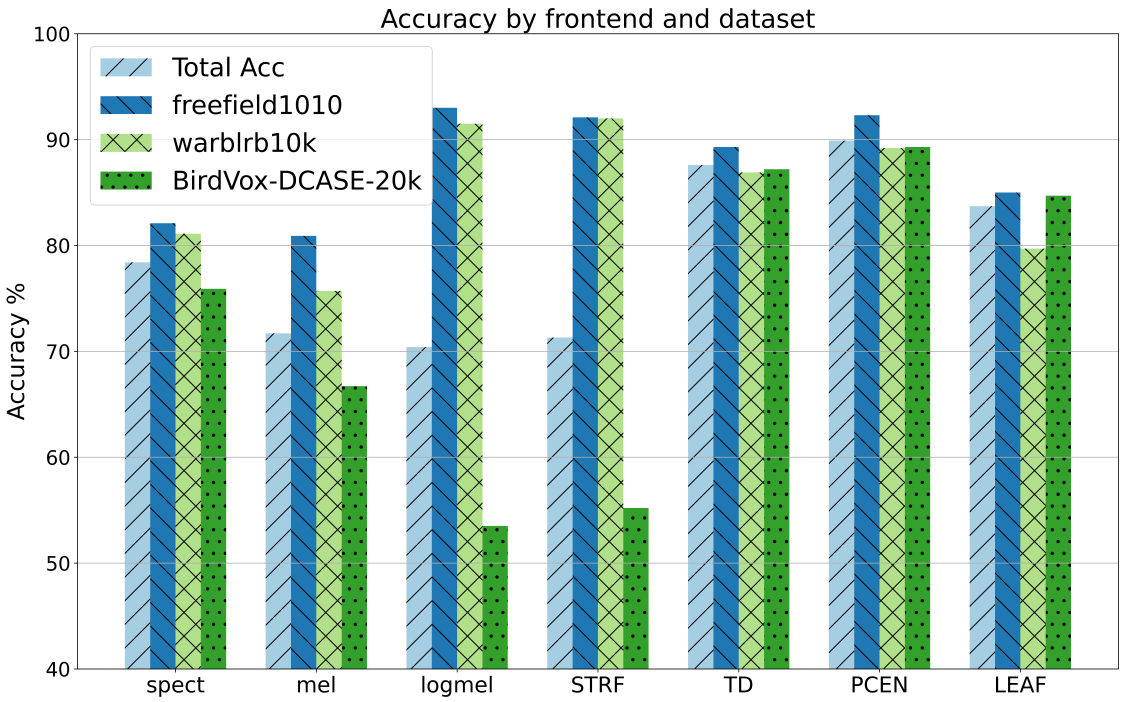

Figure 5.9: Total Accuracy on the test set, by frontend. Accuracy is also broken down by dataset.

interpretable parameters within STRF (discussed in 5.2.1) may be of interest for analysis.

Among the non-learnable frontends, uncompressed linearly scaled spectrograms (spect) achieve the highest overall performance with a test set accuracy of 78.4%. Uncompressed mel filterbank energies (mel) are the next best performer, with an accuracy of 71.7% on the entire test set. Compared to spect, mel performs similarly on *freefield1010* but exhibits lower performance on the *warblrb10k* and *BirdVox-DCASE-20k* datasets. This outcome is unsurprising, since the conversion from a full linear frequency spectrogram into a mel sepctrogram with 40 filters reduces the available information for the network.
Log-compressed mel spectrograms (logmel) exhibit the lowest performance among the traditional frontends, with a test set accuracy of 70.4%. This is mainly due to its poor performance on the *BirdVox-DCASE-20k* dataset. The logarithmic amplitude scaling applied in logmel, particularly in the presence of far-field recordings and environmental noise, potentially amplifies the noise floor and results in a lower effective signal-to-noise ratio compared to the learnable compression in PCEN, for example. However, logmel performs

well on cleaner datasets, surpassing the performance of some learnable frontends. The improved performance of logmel over spect on *freefield1010* and *warblrb10k*, despite the reduction in information, can likely be attributed to the dynamic compression achieved by logarithmic scaling of spectrogram amplitudes. As previously mentioned in Section 5.3.1, *freefield1010* and *warblrb10k* are directed, near-field recordings which contain fewer non-bird recordings than *BirdVox-DCASE-20k*. This suggests that on clean data, the benefits of a learnable compression layer may have diminishing returns.

On the whole, the learnable frontends outperform the non-learnable frontends, with the exception of STRF. STRF achieves a test set accuracy of 71.3%, similar to logmel with test set accuracy of 70.4%. Table 5.4 confirms that there is no significant difference in their results across all datasets. The minor differences in results can likely be attributed to slight variations in model weights and do not provide evidence of any performance advantages for STRF.

Among the learnable frontends, TD demonstrates the second-best overall performance with a test set accuracy of 87.6%. This indicates that frontends using learnable filterbanks can enhance performance compared to the static frequency bins of STFT based approaches. The consistent results across all datasets show strong performance even on the challenging *BirdVox-DCASE-20k data*. However, the TF-representation generated by TD is less interpretable due to the unconstrained filterbanks. While the frequency responses of the filters can be obtained through the FFT, assigning meaningful parameters such as centre frequency, bandwidth, shape becomes more challenging. The original authors of TD analyse the filter impulse responses [217], noting that they become asymmetric in a similar manner to estimates of human and animal auditory filters [176]. The authors also investigate whether the filters are analytic but do not provide specific information about centre frequencies or bandwidths. While this may not pose a problem for systems aiming for the highest performance, the unconstrained filters in TD limit researchers' ability to gain insights about the data through the network.

It is worth noting that TD is initialised to approximate an equivalent mel filterbank and utilises logarithmic compression, making it approximate to logmel. This calls for additional

comparison between the two frontends. On the entire test set, TD (87.6%) significantly outperforms logmel (70.4%). When evaluating performance on individual datasets, TD exhibits lower performance on the cleaner datasets of *freefield1010* and *warblrb10k* but surpasses logmel on *BirdVox-DCASE-20k*. Since *BirdVox-DCASE-20k* accounts for 56% of the training data, it can be hypothesised that the learned filterbank in TD is better suited to the characteristics of the *BirdVox-DCASE-20k* data compared to the other two datasets. This may explain the ability for TD to improve performance on *BirdVox-DCASE-20k* in contrast to the performance of logmel on the same data. It may also explain the decrease in performance on *freefield1010* and *warblrb10k*.

As mentioned earlier, PCEN demonstrates the highest overall performance with a test set accuracy of 89.9% and consistent results across all three datasets. PCEN offers similar performance gains to logmel on the cleaner data of *freefield1010* and *warblrb10k*. PCEN also performs well on the more challenging *BirdVox-DCASE-20k* dataset, where static logarithmic compression (as used in logmel) led to reduced performance. The success of PCEN can be attributed to the AGC and DRC, which are tuned to enhance the dynamic range of the target signal while mitigating the impact of noise on the resulting spectrogram. Note that in these experiments, a per-channel smoother (governed by the parameter s) could not be trained due to issues with model convergence, so a fixed value for s is used. Making s a learnable parameter can lead to better normalisation and noise reduction. This issue was later resolved in subsequent experiments using PCEN after identifying a source of instability (specifically in the IIR filter) in the implementation. As the IIR filter is governed by s, appropriate clamping is applied to ensure the smoothing filter is always stable.

In contrast, LEAF performs less effectively with a test set accuracy of 83.7% compared to PCEN (89.9%) and TD (87.6%), despite incorporating elements from both approaches. LEAF offers the advantage of learnable filters that are more interpretable than TD as they are parameterised by centre frequency and inverse-bandwidth, as well as using a PCEN layer for compression of the resulting magnitudes. However, it performs worse than PCEN or TD on all datasets. Two potential reasons for this are the learned filterbanks or the learned lowpass filters. Given the strong performance of TD, it would be beneficial to understand why LEAF did not perform as well or better, especially considering that LEAF is

computationally less expensive. It is more likely that the parameterised filterbank of LEAF, compared to the unconstrained filterbank in TD, is the reason for this decline in performance. While the parameterised filterbanks in LEAF allow for easier analysis and interpretation of the learned filters and TF-representation, they are limited to the space of Gabor wavelets with Gaussian frequency responses. On the other hand, TD utilises unconstrained filters, and although interpreting and analysing the learned filterbanks may be challenging, the absence of constraints on filter type and shape leads to better performance.

These results indicate that learnable compression of spectrogram magnitudes yields the most performance improvements. Logarithmic compression performs well on clean data but struggles when applied to noisy data. No compression at all proves to be more consistent than logarithmic compression, while learnable compression is both consistent and performs better in most circumstances. The usage of a learnable compression layer such as PCEN is a general recommendation for any audio tasks requiring the analysis of acoustic scenes. Learnable filterbanks, like the one used in TD, can enhance performance but to a lesser extent than learnable compression. However, when comparing PCEN vs. LEAF, learnable filterbanks seem to have reduced performance. The complexity of bird audio, especially in a species agnostic task, may pose challenges for learning filterbanks that are constrained to a specific space (such as the space of Gabor functions in the case of LEAF). While this constraint is imposed for interpretability, it negatively impacts performance on challenging data.

Additionally, examination of the learned filterbanks in both TD and LEAF indicate that the learned filters do not differ substantially from initialisation. Considering the bioacoustics task they were evaluated on, one would anticipate significant deviation from the initial mel scale. However, the observation that the filterbanks remain largely unchanged is not exclusive to this study and will be discussed in more detail in the next section of this chapter.

5.4 The Filterbank Initialisation Problem

A recurring finding, independently reported in multiple studies on learnable filterbanks [24, 111, 166, 168, 216], reveals that the filters after training do not differ substantially from

their initialised values. This is not what one might intuitively expect of learnable filterbanks that are optimised for a certain domain and task. One would expect a bespoke filterbank, tailored for the task and training data. As discussed earlier in Section 5.2, learnable frontends utilising learnable filterbanks are commonly initialised based on a static filterbank, often one linearly spaced on the mel scale. This initialisation choice is well-founded, given the prevalence and psychcoacoustic motivation of the mel scale, particularly in speech-related tasks such as keyword spotting, speaker identification and emotion recognition [166, 216].

It may be tempting to assume that if the learned filters closely resemble their initialisation (e.g. mel) then the initialisation was already well-suited for the task (e.g. speech recognition). Although this may hold true for certain tasks, it does not explain the same phenomena occurring with different filterbank initialisations, whether pyschoacoustic (such as the bark scale) or non-psychoacoustic (a linear initialisation). Moreover, there is no reason to believe that the mel scale is optimal [66], even on tasks involving human speech. This observation is likely an optimisation problem similar to the sensitivity of EM algorithms to initial values [15, 123], leading to convergence towards local optima. The initial representation can be seen as 'good enough', causing the filterbank to converge on a local optima with minimal deviation from the initialised parameters.

Motivated by the above, the objective of this study was to answer the following question: Are modern learnable frontends with trainable filterbanks sensitive to filterbank initialisation? To accomplish this, a systematic study of the sensitivity of learnable filterbanks to their initialisation is carried out, involving experiments on two distinct audio tasks: VAD and BSID. These tasks and domains complement each other as the frequency range of bird vocalisations ($\sim$ 800 Hz – 8 kHz) significantly differs from that of human speech (most energy concentrated around 300 Hz – 3.4 kHz), anticipating differences in filterbank learning.

By evaluating two tasks using two different datasets and network architectures, these experiments can verify whether this phenomena is task, dataset or model architecture specific. Four initialisation strategies are explored for both tasks, including an intentionally sub-optimal *Random* initialisation for reference purposes. For each task, the differences between the initialised and final filters are evaluated.

Some of the same studies [24, 166, 216, 217] which report a lack of learning have been dismissive of this lack of learning in filterbank based frontends. However, as in any optimisation problem, it is crucial to investigate the sensitivity to initialisation to avoid sub-optimal solutions. This section presents the experiments designed to characterise and quantify this lack of learning. The Jensen-Shannon distance is employed and additional analysis of the final filterbanks is conducted. This work was originally published in [5]. The main novelty of the study lies in the detailed quantification and interpretation of the differences between initialised and final filters for multiple initialisation strategies, as well as determining whether each initialisation led to a local optima.

5.4.1 Frontend initialisation

In this study, four different initialisation strategies are utilised. Two of these strategies are based on psychoacoustic scales, namely *Mel* [135] and *Bark* [198]. Both *Mel* and *Bark* are well suited to human speech [40, 135, 198]. However, when it comes to bird audio, previous research does not provide a consensus [184] on the preferred use of *Mel* or *Bark* compared to other scales. Some studies favour *Mel* [65], while others find *Mel* to be sub-optimal for bird audio [35]. The remaining two initialisation strategies used in these experiments are referred to as *Linear* and *Random*. The *Linear* initialisation is considered sub-optimal for both tasks [65, 88], while the *Random* initialisation is intentionally designed to be sub-optimal for both tasks.

For *Mel* and *Bark* strategies, the centre frequencies of the filterbank are linearly spaced according to their respective scales. The bandwidth parameter $\sigma_{n_{bw}}$ is set to match the *Full-Width at Half Maximum (FWHM)* of an equivalent triangular filter, corresponding to the -3 dB point. In the *Linear* initialisation, the centre frequencies are also linearly spaced, and bandwidth is constant per filter, with $\sigma_{n_{bw}}$ set in the same manner as before.

In the *Random* initialisation, the centre frequencies of the filters are determined by uniform sampling of valid frequency values within the range of 80 Hz – 8000 Hz. In order to cover the entire desired frequency range, these centre frequencies are sorted, and $\sigma_{n_{bw}}$ is set to ensure the bandwidth covers at least the FWHM of the neighbouring filters. It is important

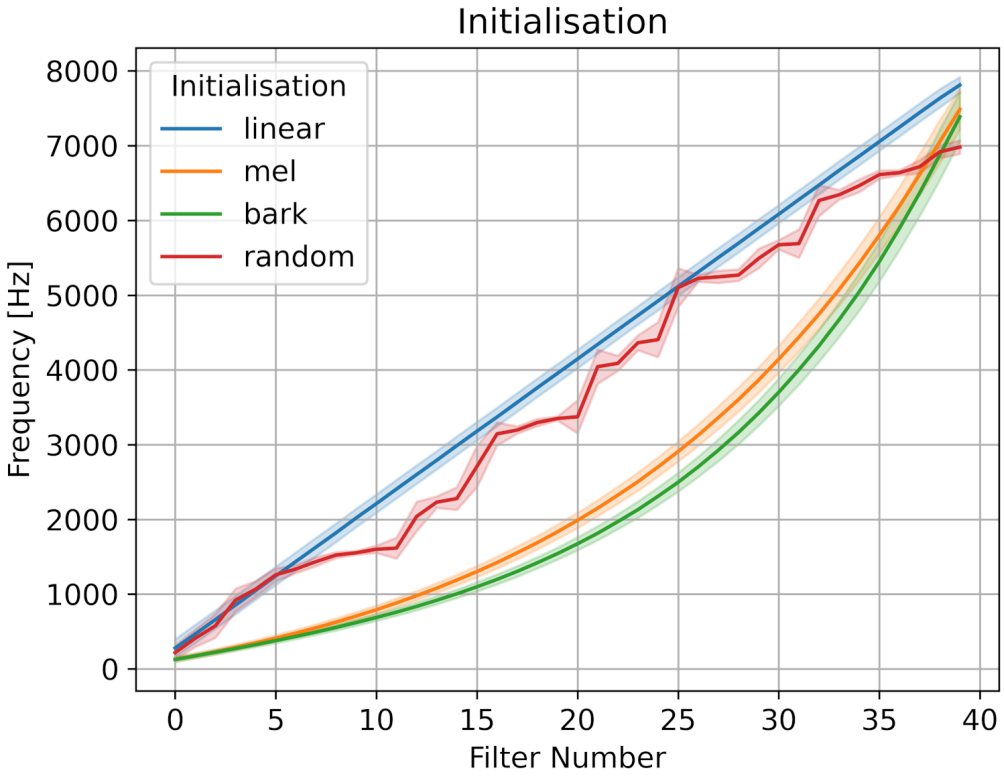

Figure 5.10: The frequency response of each filterbank initialisation. Centre frequency is represented by the solid line and bandwidth by the shaded area. In this study four initialisation types are employed: 'linear' (equally spaced, constant bandwidth), 'mel' & 'bark' (psychoacoustic pitch scales) and 'random' (ordered by frequency).

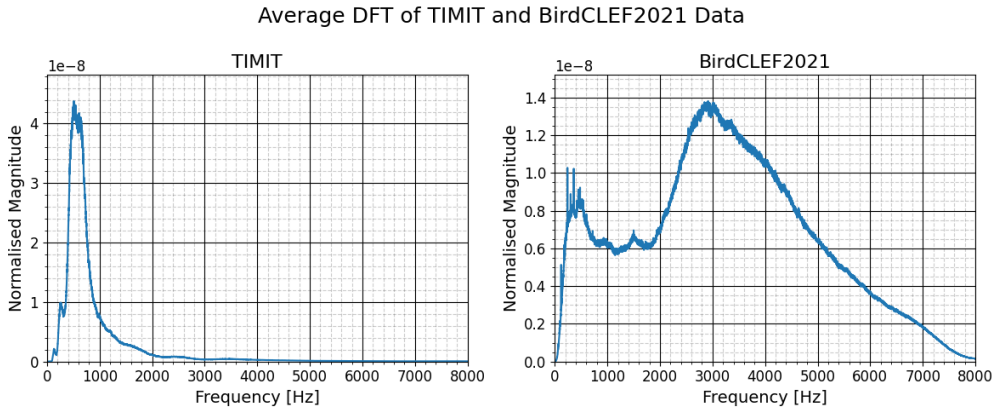

Figure 5.11: Average DFTs of all audio in both the TIMIT [59] and BirdCLEF2021 [84] datasets reveal the difference in frequency distribution between them. In TIMIT, the majority of information lies below 3 kHz, while in BirdCLEF2021, the information is more broadband with a large portion situated between 2 kHz – 5 kHz.

to note that in later chapters of this thesis, this constraint of having a well-ordered in frequency filterbank is relaxed when initialising the *Random* filterbank, allowing for a more 'random' initialisation with no requirement of covering the entire frequency range or that the filters be well ordered. This will be mentioned in the relevant section (Section 6.1.3). To ensure reproducible results, the same random seed is used in all relevant experiments. Figure 5.10 illustrates the filterbank for each initialisation strategy in the frequency domain.

5.4.2 Experimental Setup

To verify that the filterbank initialisation problem is not task, dataset or architecture specific two distinct audio tasks are used in this analysis: Voice Activity Detection (VAD) and Bird Species Identification (BSID). These tasks differ in several aspects. The VAD task is a balanced binary classification task involving near-field human speech with additive noise, whilst the BSID task is an imbalanced multiclass classification task using far-field recordings of various bird species with in-recording noise. A detailed description of both tasks is provided below, and Table 5.5 offers a summary of the tasks. Different model architectures are employed for each task, and the signals of interest have vastly different characteristics. Figure 5.11 illustrates the difference in frequency content in the two datasets, showing the average DFT of all files in each dataset and is indicative of frequencies the learned filters

should move towards. It is evident that in the VAD task, using TIMIT [59] data, most of the frequency content lies below 1 kHz, with very little frequency content beyond 3 kHz. In contrast, for the BSID task, using data from BirdCLEF2021 [84], there is a broader spectrum of frequency content primarily between 2 kHz and 5 kHz.

The frontends are configured in a manner similar to those outlined in Section 5.3. Throughout the experiments, 40 filters are utilised in the learnable frontend. Regardless of initialisation, these filters aim to encompass the frequency range spanning from 80 Hz – 8 kHz. As a baseline, log-mel spectrograms with 40 mel filters (Static Log-Mel) are used, also spanning the range of 80 Hz – 8 kHz.

While the data used in Section 5.3 has a sampling rate of 44.1 kHz, all the data in the current experiments is sampled at 16 kHz. Given that the filterbanks in Section 5.3 covered the frequency range between 500 Hz – 16 kHz, this means there is half as much frequency content in the audio of the current experiments compared to Section 5.3. Nevertheless, we use 40 filters in the learnable frontend, as this is the default number of filters used in LEAF [216]. Furthermore, we utilise a window length of 10 ms in both the learnable frontend and the log-mel baseline, with an overlap of 75% (i.e. a hop length of 2.5 ms).

Figure 6.2 displays example TF-representations for each initialisation method using portions of audio from both tasks. It is important to note that in Chapter 6, the constraint on the *Random* initialisation being well ordered is relaxed, resulting in a TF-representation that differs from how it would appear in these experiments.

Fixed filterbank models are trained for each initialisation using eLEAF. In these experiments, the PCEN and learnable low-pass filter layers remain trainable, while only the filterbank layer is fixed. Notably, the implementation of PCEN has been updated to enable training of all parameters per channel, whereas in the experiments presented in Section 5.3, the parameter s was fixed. This fully learnable PCEN layer is stable and allows for model convergence.

The method used to quantify the difference between learned and initial filterbanks, the Jensen-Shannon distance, is also detailed below in Section 5.4.2. This metric enables the quantification of the sensitivity of learnable filterbanks to their initialisation.

Voice Activity Detection (VAD)		
Dataset	**Name**	TIMIT [59]
	Sample Rate	16 kHz
	No. Speakers (Train)	630
	No. Speakers (Test)	168
	Sentences per speaker	10
	Test Set	Dedicated
	Normalisation	-6 dBFS
Classifier	**Model**	STA-VAD [100]
	Initial Learning Rate	0.001
	Optimiser	ADAM [90]
	Scheduler	Cosine Annealing [112]
	Metrics	F1, AUC

Bird Species Identification (BSID)		
Dataset	**Name**	BirdCLEF2021 [84]
	Sample Rate	16 kHz (Resampled)
	No. Species	397
	Recordings per species	Variable
	Test Set	Hold-out (15%)
	Normalisation	-6 dBFS
Classifier	**Model**	EfficientNet-B0 [190]
	Initial Learning Rate	0.001
	Optimizer	ADAM [90]
	Scheduler	Cyclic
	Metrics	F1, Acc.

Table 5.5: Details of dataset and classifier for the VAD and BSID tasks.

Voice Activity Detection (VAD)

For the VAD task, the TIMIT corpus [59] is utilised in conjunction with the Spectro-Temporal Voice Activity Detection (STA-VAD) model [100]. TIMIT provides a 'clean' corpus with separate train and test speakers sampled at 16 kHz, and is easily augmented with additive noise. STA-VAD is noise-robust and lightweight VAD system proposed by Lee et al. in 2019, incorporating both spectral and temporal attention. Similar to [100], the class imbalance between speech and non-speech is addressed by including additional silence before and after each utterance. The duration of this silence is randomly chosen to be between 0.5 seconds and 1 second.

The clean corpus is augmented with additive noise (between -10 dB and 30 dB) from the MS-SNSD dataset [153], which consists of fourteen noise types, including both stationary and non-stationary noise. The MS-SNSD noise dataset also provides test data with the same noise categories, which is used to augment the test set of TIMIT. During training, the SNR of the input signal is adaptively adjusted. This is implemented through a callback which increases the maximum allowed SNR by 5 dB (starting from -10 dB) when the validation loss during training plateaus, following a similar approach to the PyTorch *ReduceLROnPlateau()* callback. Each epoch involves shuffling the data and equally splitting it based on the available SNR values. The data are then augmented with randomly selected noise at the corresponding SNR values. The power of the added noise is calculated based only on segments containing speech, excluding silence. Both TIMIT and MS-SNSD are sampled at 16 kHz.

The STA-VAD model is trained using a 90% training and 10% validation split. The training regime includes the utilisation of a cosine annealing learning rate scheduler [112], which is set to restart approximately when new SNR values are introduced. During the validation stage, the noise levels are randomly selected from the currently available SNR values, following a uniform distribution. The test data is from the TIMIT test set and augmented with additional silence and additive noise at a SNR of 15 dB. As mentioned earlier, the additive noise in the test set is randomly selected from all noise categories in the MS-SNSD test set. In line with [100, 166, 216], the experiments report accuracy and AUC.

Bird Species Identification (BSID)

In the BSID task, the publicly available training dataset from BirdCLEF2021 [84] is used in conjunction with EfficientNet-B0 [190]. Both the dataset and model were chosen to allow for a comparative analysis with the results from [166]. EfficientNet-B0 has also been used extensively in the original LEAF implementation paper [216] for evaluating various tasks.

The BirdCLEF2021 dataset consists of high-quality focal recordings of variable lengths sourced from Xeno-Canto, an online repository of crowd-sourced bird recordings. These recordings have been normalised to -6 dBFS. As Xeno-Canto is a crowd sourced repository, the original sampling rates of recordings vary, with many recorded at 44.1 kHz. Additionally the recordings are in a variety of codecs, such as MP3 or Ogg Vorbis. All recordings are resampled and converted to 16 kHz WAV files to match the specifications of TIMIT [59]. There are 397 species present in the dataset, from across North & South America. There is a significant class imbalance, with 12 species having 500 recordings while 9 species have fewer than 25 recordings.

Due to the unavailability of a test set for the BirdCLEF2021 dataset, a test set is generated by subsampling the existing data. To this end, a 70:15:15 dataset split, with consideration for the class imbalance, is employed for training, evaluation, and testing. In these experiments, both accuracy (as utilised in the evaluation of eLEAF [166]) and F1-Score (official metric of BirdCLEF2021) are reported.

Evaluation of Frontend Sensitivity using the Jensen-Shannon distance (JSD)

To assess the sensitivity of learnable filterbanks to initialisation, it is necessary to employ a suitable metric that quantifies the difference between the initialised and final filterbanks. In this study, the Jensen-Shannon distance (JSD) is utilised as the metric for quantifying this difference. The JSD, defined by Equation 5.29, offers a means of measuring the similarity or dissimilarity of two probability distributions. It is derived from the Kullback-Leibler divergence (defined by Equation 5.30), and compares each distribution P and Q to a mixture distribution M. The JSD is a true metric [50] that satisfies all related axioms: a distance of 0 when comparing a distribution to itself, the positivity axiom, the symmetry axiom and the

triangle inequality. When a logarithmic base of 2 is employed in Equation 4.32, the JSD is bounded within the range of $[0, 1]$.

$$D_{\mathrm{JS}}(P, Q) = \sqrt{\frac{1}{2}\left[D_{\mathrm{KL}}(P||M) + D_{\mathrm{KL}}(Q||M)\right]} \tag{5.29}$$

$$\text{Where, } M = \frac{P + Q}{2}$$

$$\text{And, } D_{\mathrm{KL}}(A||B) = \sum_{x} A(x) \log\left(\frac{A(x)}{B(x)}\right) \tag{5.30}$$

In LEAF, the frequency response of each filter is a sampled Gaussian kernel and can be interpreted similarly to a probability distribution. Typically used to compare between a ground truth distribution and a distribution of simulated values, in this context it compares between an initial distribution and a final distribution. As the JSD is bounded, $D_{\mathrm{JS}}(P, Q) = 1$ means that the initial and learned filters are completely dissimilar to each other. However, an issue arises in the subsequent analysis of the filterbanks when dealing with filters having narrow bandwidths. Even slight changes in centre frequency can result in significant distance variations, despite the overall position and shape of the filter remaining relatively unchanged. To address this challenge, the final learned filters are also evaluated in terms of changes in centre frequency and bandwidth relative to their initialisation.

5.4.3 Results & Discussion

Table 5.6 reports the performance of each task on their respective test set, although these results are not the focus of this study. Instead, they demonstrate that each model has been trained successfully and allow for comparisons between fixed filterbanks and learnable filterbanks. It is evident that fully learnable frontends outperform their fixed filterbank counterparts. The fixed filterbank learnable frontends also exhibit better performance than the log-mel spectrogram baseline features, except for the *Random* initialisation, which is intentionally sub-optimal. In the VAD task, the AUC metric is an exception where fixed filterbanks outperform learnable filterbanks. Although learnable filterbanks with *Random* initialisation show improvement compared to their fixed counterparts, they perform worse

	VAD				BSID			
	F1-Score		AUC		F1-Score		Acc.	
Filterbank	*Fixed*	*Learn*	*Fixed*	*Learn*	*Fixed*	*Learn*	*Fixed*	*Learn*
Linear	0.841	**0.862**	0.839	**0.858**	0.663	**0.674**	0.716	**0.734**
Mel	0.826	0.858	0.778	0.840	0.660	0.668	0.714	0.718
Bark	0.834	0.860	0.816	0.846	0.660	0.665	0.708	0.716
Random	0.802	0.807	0.765	0.759	0.653	0.662	0.681	0.715
Log-Mel	0.847	—	0.881	—	0.646	—	0.697	—

Table 5.6: Results on hold-out test set for VAD and BSID tasks. Includes results using learnable frontends with fixed filterbanks (*Fixed*) and using learnable frontends with learnable filterbanks (*Learn*). Results between each *fixed*/*learn* pair are statistically significant. Best results for the learnable frontend are marked in **bold**.

than learned filterbanks initialised with other strategies. This can be attributed to the sub-optimal nature of the random initialisation, which hampers learning and consequently affects the final performance of the model.

The most significant performance improvements from static baseline features to learnable features are observed in the BSID task. This can likely be attributed to the PCEN compression layer, as discussed in Section 5.3.4 and in the accompanying paper [3]. Unlike a static logarithmic compression, which performs well on cleaner data with subjects in close proximity, PCEN enhances the dynamic range of the signal and reduces noise, particularly in far-field and noisy recording conditions.

Regarding the VAD task, performance improvements when using learnable filterbanks are more evident. Previous work [65, 88] suggests that a fixed *Linear* filterbank is sub-optimal, with some studies favouring the *Mel* initialisation for both tasks. However, in both fixed and learnable scenarios, the *Linear* initialisation performs best for both VAD and BSID tasks. A generalisation cannot be made that *Linear* is the best initialisation to use with learnable filterbanks. However, it can be speculated that by not prioritising certain frequencies in the spectrum, as *Mel* and *Bark* do, the model as a whole can learn more from the data. This may especially be true in the case of the bird audio task as many bird species vocalise above 1000 Hz, below which most of the energy in the *Mel* and *Bark* filterbanks are contained. This speculation holds true whether the filterbank is further tuned in training, or is fixed.

In the VAD task, the best result was achieved using learnable filterbanks with linear initialisation (F1-Score 0.862, AUC 0.858). Whilst a direct comparison is challenging due to a difference in the testing dataset, our performance aligns with the performance reported in [100] (Mean AUC 0.886). This indicates that the model was trained correctly, with differences arising due to the different test sets. The AUC of the baseline log-mel spectrogram is higher than that of the learnable frontend, with the log-mel spectrogram achieving an AUC of 0.881, compared with the next highest AUC of 0.858 in the learnable *Linear* filterbank. This suggests that the model using the log-mel spectrogram baseline may generalise better, but in terms of classification performance, the learned *Linear* filterbank performs best.

In the BSID task, again the best results are achieved using trainable filterbanks with linear initialisation (F1 0.674, Acc. 0.734). Direct comparison of accuracy is possible, with the performance of this study being in line with [166], who reported an accuracy of 0.722 on this task. Once again, this indicates that the model was trained correctly, and that investigation into the state of the filterbanks can be undertaken with confidence.

The more pertinent findings of this study can be seen in Figures 5.10, 5.12 and 5.13. Figures 5.10 and 5.13 show the frequency responses of each filterbank before and after training, while Figure 5.12 depicts the filterbank movement from initial values over time, using the JSD. The triangle inequality property of the JSD allows for the plotting of this figure. In Figure 5.12, for the VAD task there is very little movement from the *Linear* and *Bark* initialisations (0.07 and 0.10 respectively), with no filters moving substantially. The *Mel* initialisation (0.13) displays movement in the lower frequency filters, showing a shift to lower centre frequencies while maintaining the same bandwidth (as seen in Figure 5.13, although the change is subtle). This shift results in a large distance due to the small bandwidth of the low frequency filters, any change in centre frequency for low bandwidth filters represents a substantial change in distance when using the JSD. The intentionally sub-optimal *Random* initialisation shows more movement, with mean distance of 0.23. Despite the greater movement compared to the learned *Linear* filterbanks, other initialisation strategies do not provide as good a solution. Contrasting these distances with the final learned filters in Figure 5.13, we observe that the overall state of the filterbanks have not

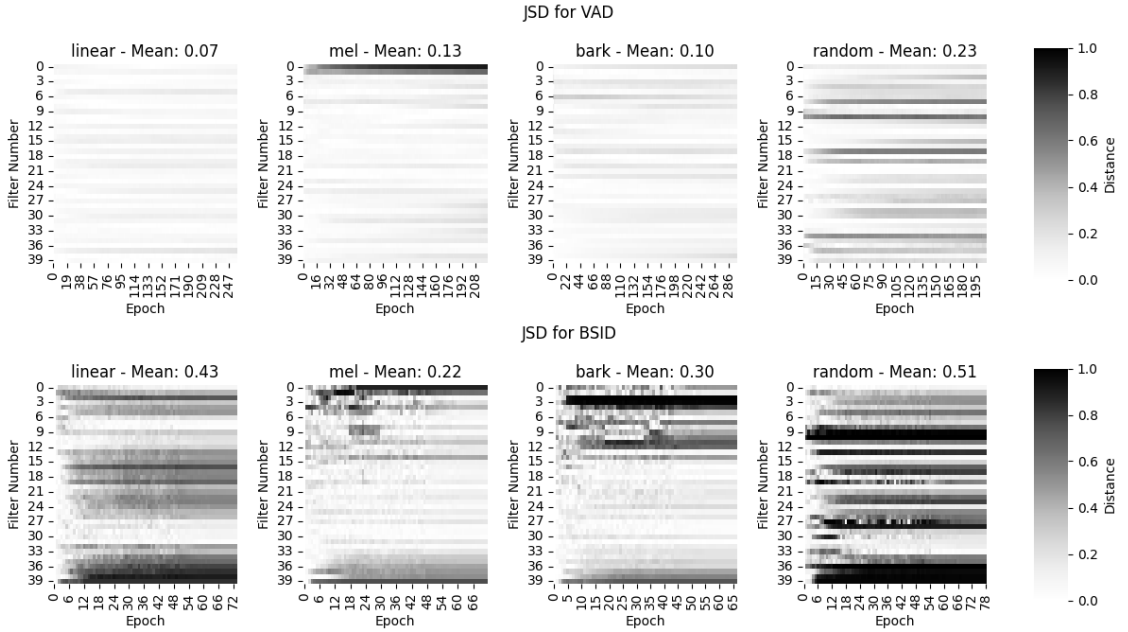

Figure 5.12: Jensen-Shannon distance of each filter from its initialisation for the VAD and BSID tasks, by initialisation strategy. The mean of the final distances is also shown in each plot's title.

changed substantially from their initialisation, with no sign of movement towards a general, optimum solution.

Comparing these findings to the BSID task, in Figure 5.12 we see more movement between initial and final filters compared to the VAD task. The *Mel* and *Bark* initialisations move least (0.22 and 0.30 respectively), both showing movement in the lower and higher frequency filters. The *Linear* initialisation (0.43) has more movement than either of the psychoacoustic initialisations, with movement across most frequency channels. Similar to the VAD task, the *Random* initialisation shows the most movement (0.51). However, when contrasting again to Figure 5.13, we do not see change in the overall state of the learned filterbanks with the exception of *Random*, which moves towards a linear-like state. Similar to VAD, although the learned filterbanks in the BSID task exhibit some movement they do not achieve similar performance to the best performing filterbank (*Linear* initialisation). If the optimisation strategy was functioning as intended, all learned filters would achieve similar performance.

This study demonstrates the sensitivity of learnable filterbanks to initialisation, and quantifies

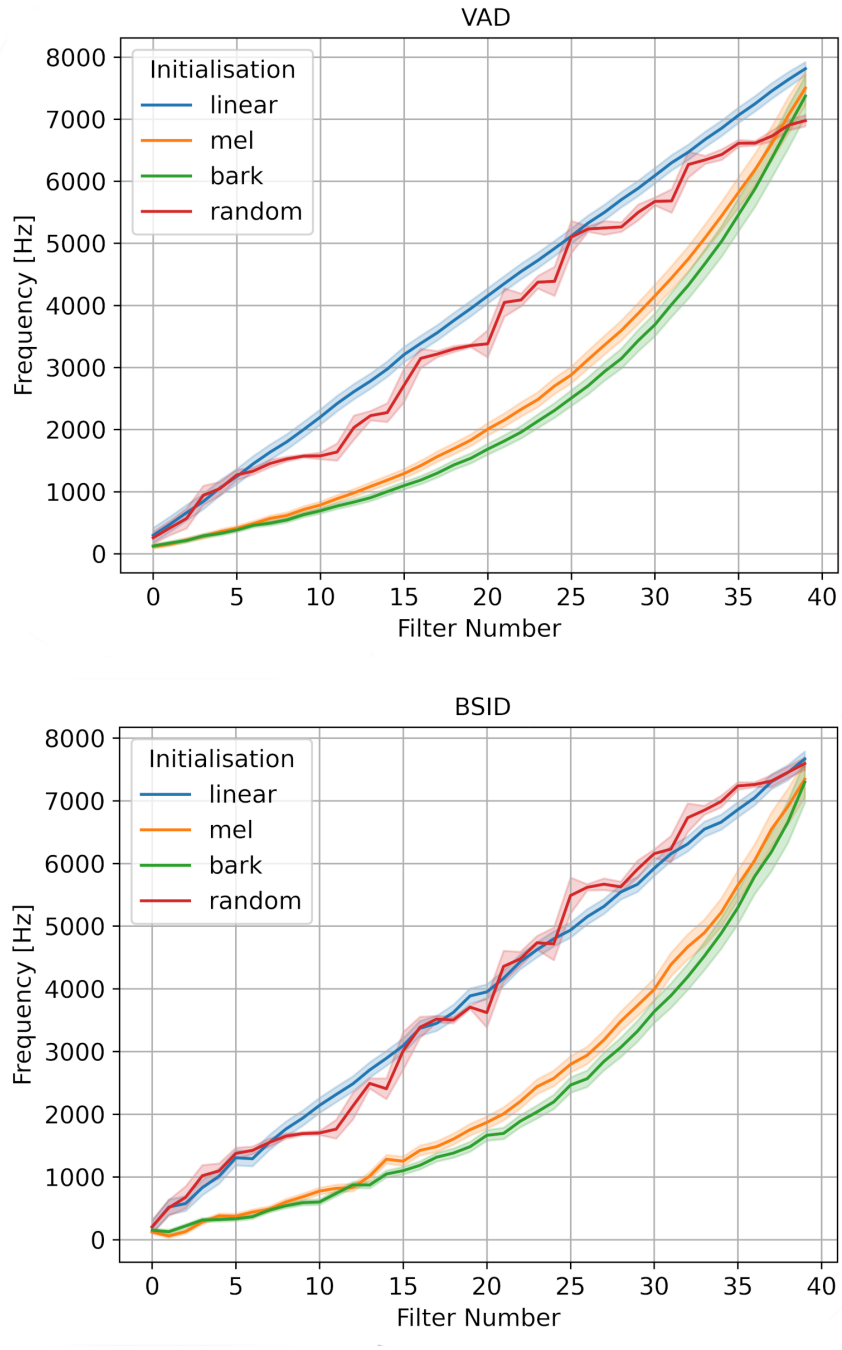

Figure 5.13: The frequency response of the learned filterbanks after training on each task (VAD and BSID). Centre frequency is represented by the solid line and bandwidth by the shaded area.

the change from initialisation in the final learned filters. The findings indicate a lack of learning in learnable filterbanks, which is consistent across two different tasks and four initialisation strategies. While performance can be improved with learnable filterbanks, it comes at additional computational cost. These results have implications in considering the trade-off between training time and model utility.

This section introduced a methodology for quantifying the lack of filter movement in learnable filterbank-based audio frontends, offering novel insights into their shortcomings. However, for learnable filterbanks to be merited, they must offer reliable performance increases. Ideally, learnable filterbanks should move from their initialisation to a set of optimal filters. This should happen with any reasonable initialisation, such as *Mel*, *Bark* or *Linear*. The inconsistency in the performance of the learned filters, coupled with the lack of movement from initial filterbank values, demonstrates a shortcoming in the overall optimisation strategy. Chapter 6 builds upon this work, with the development of tangible mitigation strategies to these shortcomings. Furthermore, it applies the insights gained from this chapter and the first section of Chapter 6 to integrate learnable frontends into the FSL system developed in Chapter 4.

Chapter 6

Mitigating the Filterbank Initialisation Problem and Returning to FSL

Chapter 5 demonstrated that learnable frontends can offer performance increases over static, non-learnable counterparts despite some limitations. In particular, the use of learnable compression, such as PCEN [211], substantially enhances performance compared to using logarithmic compression, or no dynamic range compression at all (refer to Table 5.3 and Figure 5.9). Learnable frontends incorporating filterbanks also have the potential to improve performance (see Table 5.6), however they are sensitive to their initialisation. This sensitivity manifests in two ways: firstly, the filters exhibit reluctance to move from their initialisation in terms of centre frequency or bandwidth; secondly, differing performance across initialisations, even after training. Ideally, optimisation of the filterbank should yield consistent performance regardless of the initialisation, within reasonable limits. This chapter proposes two mitigation strategies for the filterbank initialisation problem, along with the application of both learnable compression (PCEN) and an 'all-in-one' learnable frontend (eLEAF) to the prototypical network discussed in Chapter 4. These proposals aim to address the filterbank initialisation problem and further explore FSL as a means of implementing BAD.

This thesis chapter is separated into two sections: Firstly, Section 6.1 outlines two strategies to counteract the filterbank initialisation problem as seen at the end of Chapter 5 (Section 5.4). Section 6.1.1 and Section 6.1.2 detail the two suggested modifications to the training strategy. Section 6.1.3 describes the experimental setup, while Section 6.1.4 presents and discusses the results of these experiments. Subsequently, Section 6.2 revisits FSL, this time incorporating learnable frontends. Section 6.2.1 briefly discusses the advancements in FSL for bioacoustics since the 2021 DCASE challenge. Section 6.2.2 recaps the experimental setup from Chapter 4 and finally Section 6.2.3 presents and discusses the results of these experiments.

6.1 Mitigation Strategies for the Filterbank Initialisation Problem

Section 5.4.3 demonstrated that learnable frontends using learnable filterbanks exhibit sensitivity to their initialisation. The learned filters do not differ substantially from their initialisation, indicating a lack of learning. The choice of initialisation impacts performance, and the learned filters fail to compensate for this choice. While learnable filterbanks do enhance performance, these increases should be reliable and not contingent on their initialisation, the filters should move from any reasonable initialisation to a set of optimal filters. Determining whether the learned filters are optimal remains challenging; nevertheless, regardless of their intialisation, the learned filters should exhibit similarity post-training. Additionally, observed changes in the frontend appear to be model-dependent; the learned filterbanks with EfficientNet-B0 [190] exhibit more deviation from their initialisation compared to those in the STA-VAD [100] network. This points to a shortcoming in the optimisation strategy, as not only do the filters not move from their initialisation, but their movement is both data and model dependent.

These shortcomings lead to the belief that the best optimisation strategy for the backend may not necessarily be the most effective for the frontend. Two improvements to the training method are proposed, which aim to mitigate some of the effects of the initialisation problem. These two improvements are: *alternate training* (alternating training between the

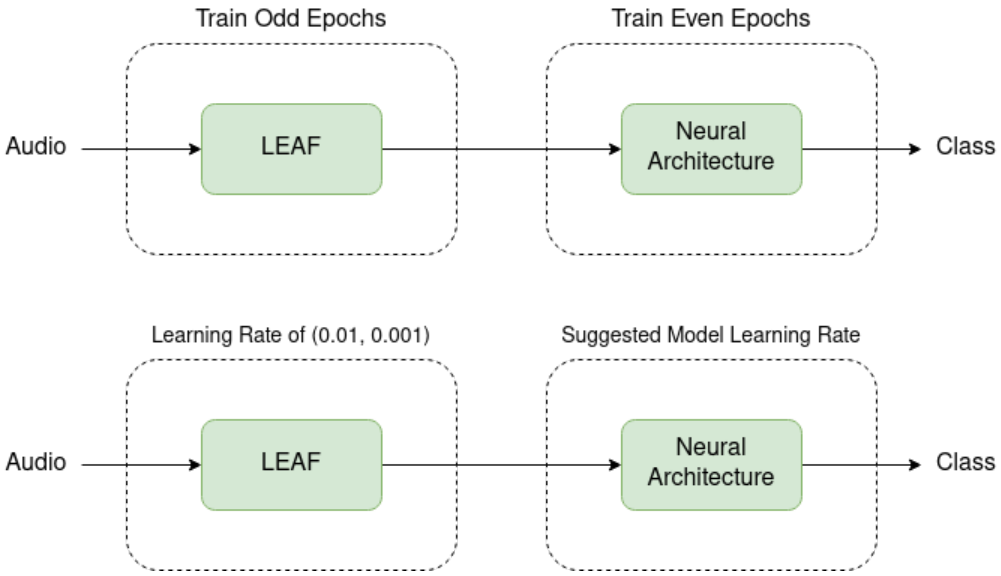

Figure 6.1: The proposed mitigation strategies involve alternate training (top) and using separate optimisation (bottom) of the frontend and backend. Alternate training involves optimising only one section of the model at a time for a specified period (e.g. one epoch). Separate optimisers involve the usage of two optimisers. This can include the use of different optimisation algorithms and learning rates; however, in these experiments, only the learning rates differ. The following experiments also employ a combination of both strategies.

frontend and the backend), and *separate optimisation* of the frontend and backend. These methods are depicted in Figure 6.1, with further details provided in Section 6.1.1 and Section 6.1.2 below.

6.1.1 Alternating Training

Alternating training is inspired by alternating optimisation methods. Bezdek and Hathaway [12, 13] formulate alternating optimisation as an iterative procedure, which minimises some function $g(\theta)$ jointly over all parameters θ by alternating minimisation over non-overlapping subsets of θ. Applying this principle to a deep learning context, we wish to minimise the loss function used to train the model, $L(f_\theta(\mathbf{X}), \hat{y})$, jointly over the model parameters θ by alternating minimisation over the parameters of the backend θ_b and the parameters of the frontend θ_f. These parameters are non-overlapping and form the complete set of parameters in the model $\theta = \{\theta_f, \theta_b\}$. Alternate training has some differences to alternating optimisation, but is inspired by this idea of minimising parameters separately.

Alternating optimisation-based methods have been deployed in deep learning [219], particularly in the problem of Blind Super-Resolution [75, 102]. In some cases, it is utilised as an alternative to gradient descent-based optimisation [219], whereas others utilise gradient descent-based minimisation in conjunction with an alternating optimisation-based framework. Gradient-descent-based optimisation is used in the training of our models. Luo et al. [75] used this hybrid approach in their work on super resolution of images, optimising sections of their model alternately such that they work in tandem towards an optimum. This is the most similar to our alternate training method.

This method allows the model to initially optimise the backend layers based on the frontend's initialisation, enabling the backend to learn certain features and a classifier. After this initial learning phase, the backend is frozen, and the frontend layers are subsequently optimised to enhance the TF-representation of the signal. This process is repeated according to a predefined schedule (e.g., optimising each set of parameters on alternating epochs), and training concludes upon reaching convergence. In this way, it is hoped that the filterbank is forced to move away from it's initialisation by restricting the available parameters for loss minimisation to those within the frontend.

Practically, this alternate training approach is implemented by grouping frontend and backend parameters into separate lists. According to a specified schedule, a flag determining whether these parameters should be included in the backwards pass of the model (calculating gradients and optimising weights) is alternately set to true or false. In the following experiments, these flags are simply toggled every epoch. During even epochs (starting from epoch 0), the backend is optimised while the frontend layers remain frozen. On odd epochs, the frontend is optimised while the backend layers are frozen. This is depicted by the top section of Figure 6.1. An early stopping method is employed to check for model convergence. Although this simple even/odd toggling system is used, more intricate scheduling options could be explored. However, these alternatives are not tested in this work due to limitations in experiment runtime.

6.1.2 Separate Optimisation

Separate optimisation involves employing several optimisers to train specific segments of the model, which can be done within the same backwards pass, as opposed to alternating optimisation. The change in the filterbank between its initialisation and the end of the training stage varies between models (refer to Figure 5.12 and Figure 5.13). With the belief that a stable learning rate for the backend might be too conservative for the frontend (especially the filterbank), isolating parameters for these sections and employing separate optimisers presents a promising approach to addressing the lack of movement.

While conceptually simpler than alternating optimisation, separate optimisation provides considerable flexibility during training. In its simplest form, two learning rates are employed, with the backend's rate typically configured for less aggressive, more stable updates, and the frontend allowed to make more aggressive parameter changes. This simple case, where only the learning rate differs between the optimisers, is depicted in the lower part of Figure 6.1. However, this method allows for using different optimisation methods, differing optimisation hyperparameters, or applying different regularisation methods depending on the layer type. Separate optimisation is widely implemented in some form in many deep learning tasks and architectures, such as Generative Adversarial Networks (GAN) [149], but more typically in cases where there are separate encoder/decoder architectures.

Practically implementing separate optimisers is straightforward, facilitated by the support for multiple optimisers for different parameter groups in deep learning frameworks like PyTorch and TensorFlow. This is an easy and flexible addition to existing code and still enables running the entire model with the same learning rate and optimisation algorithm, all specified in a configuration file.

6.1.3 Experimental Setup

The tasks, models and data used in the following experiments are the same as those detailed in Section 5.4 for characterising the filterbank initialisation issue. For more details on the two tasks (VAD and BSID), datasets, and models used, refer to Section 5.4.2 and Table 5.5. The same metrics and analysis methods, including JSD and examination of learned filterbank

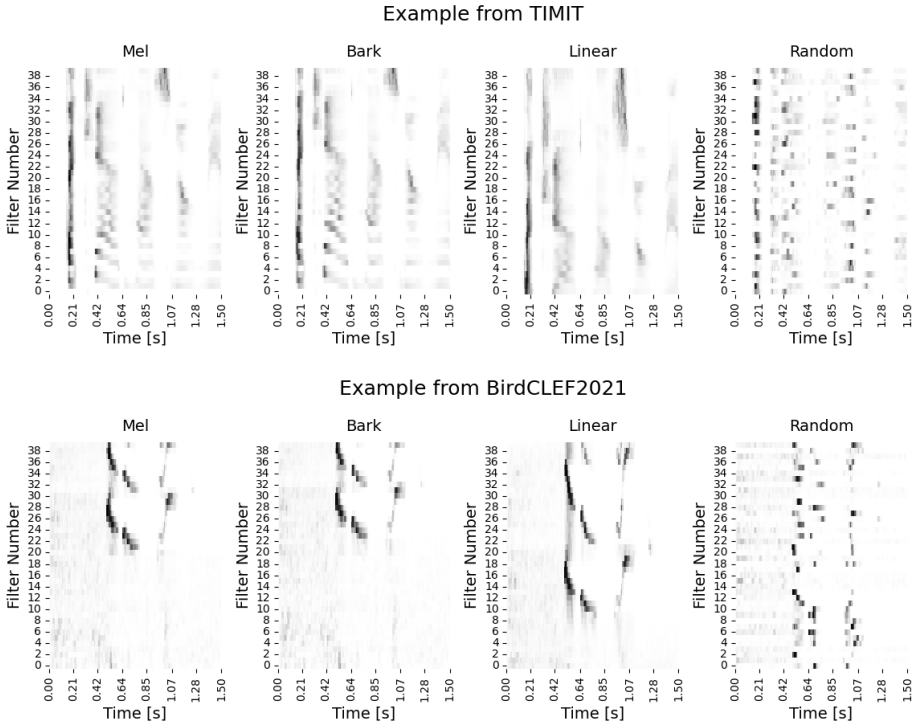

Figure 6.2: TF-representations of each initialisation method before training using portions of audio from both the TIMIT [59] (with no additive noise) and BirdCLEF2021 [84] datasets. PCEN parameters are initialised using $\mathbf{s} = 0.05$, $\alpha = 0.98$, $\delta = 2$ and $\mathbf{r} = 0.5$.

properties, are applied. The frontend employed is still eLEAF [166, 216] with a fully trainable PCEN [211] compression layer.

The experiments involve alternating training (*Alt.Train.*), separate optimisation with different learning rates (*Diff LR*), and a combination of both (*Both*) applied to the model's training. A *Baseline* method is also trialled, simply training the learnable frontends in an 'off-the-shelf' manner. These *Baseline* results include the original models trained in Section 5.4. There is an exception to this, the *Random* initialisation has been modified so that it is no longer well-ordered in frequency. Subject to some constraints on centre frequency, bandwidth and spectrum coverage, this initialisation is now more random than the previous *Random* initialisation. Consequently, the previous *Random* results from Chapter 5 have been discarded.

Figure 6.2 displays sample TF-representations for each initialisation method using portions of audio from both tasks. This enables us to observe the characteristics of the TF-representation

for each filter initialisation. *Mel* and *Bark* initialisations better capture the harmonics and formants of human speech due to more filters at lower frequencies. In contrast, *Linear* initialisation captures the high-frequency components of bird audio without any distortion or warping of the frequency axis. Additionally, the TF-representation for *Random* initialisation illustrates the absence of well-ordered centre frequencies in this initialisation.

There are 16 configurations in total, applied to both tasks, training models using each initialisation (*Linear*, *Mel*, *Bark* and *Random*) and strategy (*Baseline*, *Alt.Train.*, *Diff LR* and *Both*). Results are presented as the sample mean of a metric $\pm$ the sample standard deviation over three training runs, as in the original LEAF and eLEAF papers [166, 216]. Further analyses of the learned filters are taken from the model with the best F1-Score of the three trained models for each initialisation/strategy combination.

Alt.Train. switches parameter groups to be optimised every epoch, while separate optimisation uses the same optimiser (ADAM [90]) for both sets of parameters. The learning rate for the frontend is set to 10^{-2}, representing a $10\times$ increase over the backend learning rate. Backend optimisers are subject to learning rate schedules, as outlined in Table 5.5, and an early stopping callback halts training after 10 epochs of convergence on a minimum, as in Section 5.4.

6.1.4 Results & Discussion

This section begins by discussing model performance, focusing on the optimal initialisations and strategies for each task. Discussion is provided for each task individually, followed by overall observations. Subsequently, an in-depth analysis of the learned filters for each strategy is presented. This section concludes with some general remarks on method suitability and recommendations for filterbank initialisation.

Performance

Table 6.1 details the performance of each task for different initialisation and strategy combinations. Generally, the use of the additional optimisation strategies increases performance, especially in the VAD task. In contrast to the findings in Chapter 5, where

Initialisation	Strategy	VAD		BSID	
		F1	AUC	F1	Accuracy
Linear	Base	0.862 ± 0.003	0.857 ± 0.002	$\mathbf{0.672 \pm 0.002}$	$\mathbf{0.729 \pm 0.003}$
Linear	Alt. Train.	0.788 ± 0.004	0.788 ± 0.004	0.655 ± 0.005	0.694 ± 0.002
Linear	Diff LR	$\mathbf{0.866 \pm 0.002}$	$\mathbf{0.860 \pm 0.002}$	0.666 ± 0.004	0.714 ± 0.002
Linear	Both	0.837 ± 0.005	0.823 ± 0.002	0.650 ± 0.003	0.688 ± 0.004
Mel	Base	0.855 ± 0.003	0.839 ± 0.002	0.667 ± 0.003	0.717 ± 0.005
Mel	Alt. Train.	0.824 ± 0.009	0.823 ± 0.003	$\mathbf{0.668 \pm 0.004}$	$\mathbf{0.726 \pm 0.002}$
Mel	Diff LR	$\mathbf{0.870 \pm 0.002}$	$\mathbf{0.861 \pm 0.001}$	0.618 ± 0.005	0.643 ± 0.003
Mel	Both	0.828 ± 0.009	0.848 ± 0.004	0.503 ± 0.008	0.615 ± 0.005
Bark	Base	0.856 ± 0.004	0.844 ± 0.002	$\mathbf{0.664 \pm 0.003}$	$\mathbf{0.711 \pm 0.005}$
Bark	Alt. Train.	0.726 ± 0.005	0.711 ± 0.004	0.518 ± 0.003	0.634 ± 0.004
Bark	Diff LR	$\mathbf{0.867 \pm 0.002}$	$\mathbf{0.859 \pm 0.003}$	0.516 ± 0.003	0.630 ± 0.004
Bark	Both	0.839 ± 0.007	0.850 ± 0.001	0.658 ± 0.006	0.694 ± 0.004
Random	Base	0.833 ± 0.007	0.825 ± 0.003	0.455 ± 0.002	0.562 ± 0.002
Random	Alt. Train.	0.837 ± 0.006	0.843 ± 0.002	0.448 ± 0.004	0.553 ± 0.004
Random	Diff LR	$\mathbf{0.869 \pm 0.004}$	$\mathbf{0.872 \pm 0.002}$	$\mathbf{0.463 \pm 0.004}$	$\mathbf{0.572 \pm 0.002}$
Random	Both	0.793 ± 0.007	0.793 ± 0.009	0.430 ± 0.003	0.532 ± 0.003

Table 6.1: Results on hold-out test set for VAD and BSID tasks. The *Baseline*, *Alt. Train*, *Diff LR* and *Both* strategies are evaluated across different initialisations (*Linear*, *Mel*, *Bark* & *Random*). The values presented are the mean and standard deviation of each metric over three runs (as in [166, 216]). Best results for each task per initialisation are highlighted in **bold**. Best overall results for each task are marked in **red**.

Linear initialisation performed best on both tasks (refer to Table 5.6), this is no longer the case for the VAD task. When employing mitigation strategies in the VAD task, *Mel* and *Random* initialisations yield better results.

For the VAD task, the best F1-Score result comes from using *Mel-Diff LR* (0.870), employing a more aggressive frontend optimisation. *Random-Diff LR* produces the best results on the AUC metric (0.872). *Random* initialisation should not be considered a valid initialisation due to dependency on the seed and generation algorithm. However, these results suggest that a well-ordered initialisation might not be essential for good performance, depending on the backend model and task. Regardless of the initialisation strategy, using separate optimisers with a higher learning rate for the frontend layers consistently improves performance over the baseline model.

On the BSID task, *Linear-Baseline* yields the best results (0.672 for F1-Score and 0.729 for

accuracy). *Linear* initialisation remains the overall best performer on the BSID task. *Alt.Train.* improves the performance of the *Mel* initialisation, whereas *Diff LR* improves the results of the *Random* initialisation. Both *Linear* and *Bark* initialisations do not benefit from using either of these strategies.

In the VAD task, *Random* initialisation does not significantly impact performance. When coupled with *Diff LR*, it improves from the baseline and competes with other initialisations. Conversely, in the BSID task, *Random* initialisation performs significantly worse than other methods. Alongside the practical concerns associated with *Random* initialisation, the success of learning from the resulting TF-representation appears task and model-dependent. Considering the issues related to seeding and implementation of the random number generator and generator, it is advisable to avoid random initialisation, even if the model can learn from the TF-representation.

Overall, the use of separate optimisers with a more aggressive learning rate has improved the performance on the VAD task. Additionally, *Mel* initialisation remains a sensible starting point for tasks involving human speech. It should also be noted that with the exception of the AUC metric using a *Random* initialisation, the *Diff LR* strategy results show similar performance, varying by at most 0.008 in F1-Score and 0.006 in AUC.

Contrast this to the BSID task, where there is no agreement on whether one strategy is best. However, it seems that no mitigation strategy (i.e. *Baseline*) may be preferable. In Section 5.4.3, it was noted that learned filterbanks in the BSID task differ more from their initialisation compared to the VAD task. The effectiveness of a strategy such as *Diff LR* may also be model and data-dependent. Both *Alt.Train.* and *Diff LR* perform well on *Mel* and *Random* initialisations, while *Baseline* works best with *Linear* and *Bark* initialisations. Considering the best performance is achieved using *Linear-Baseline*, it is recommended to use a linear initialisation with bird audio.

Analysis of Learned Filterbanks

Examining the learned filterbanks in the VAD task, Figure 6.3 illustrates the filterbanks' movement over time, while Figure 6.4 presents the frequency responses of each filterbank.

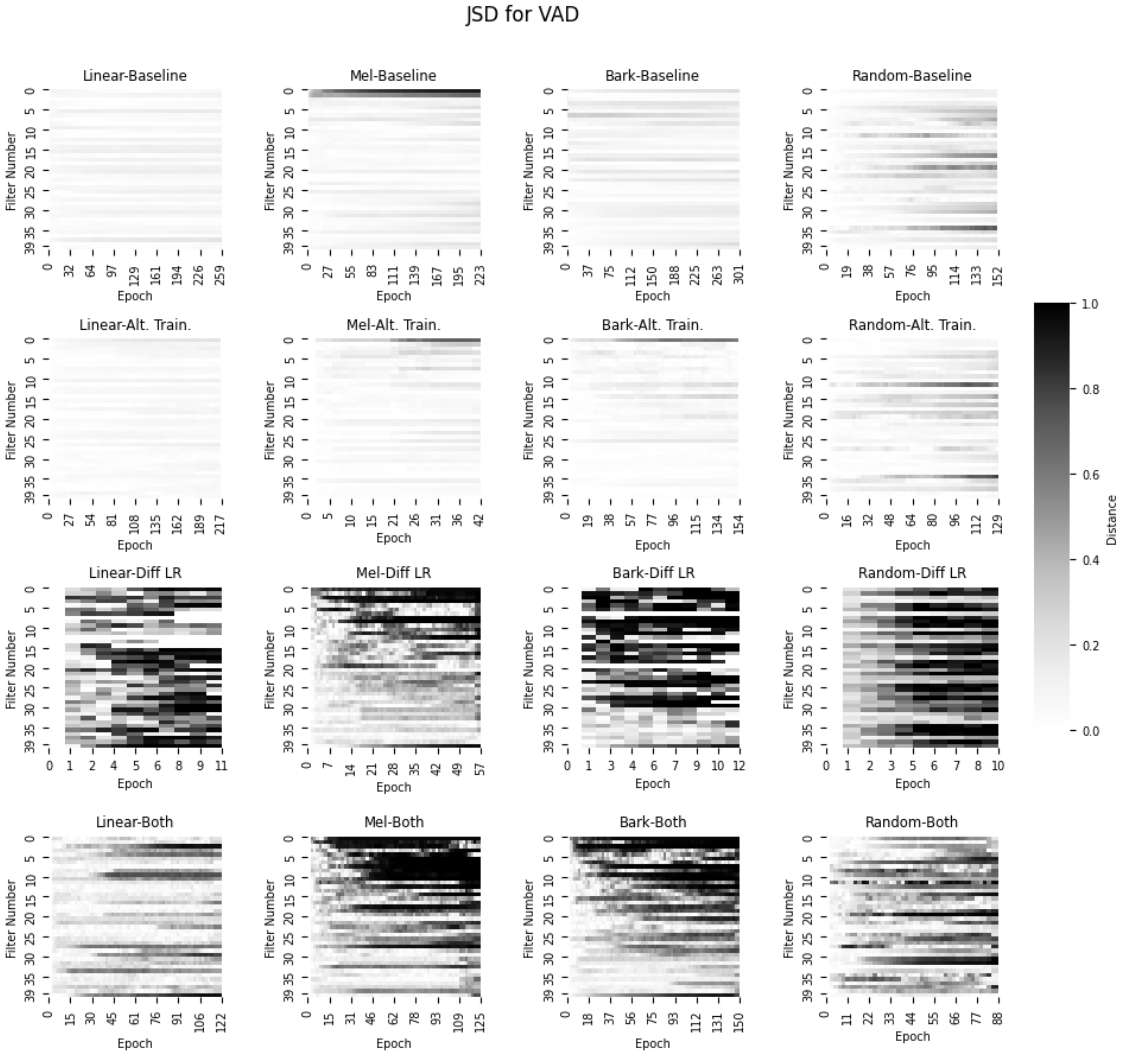

Figure 6.3: Jensen-Shannon distances of each filterbank from its initialisation for the VAD task. This figure is ordered by initialisation horizontally, and by training strategy vertically.

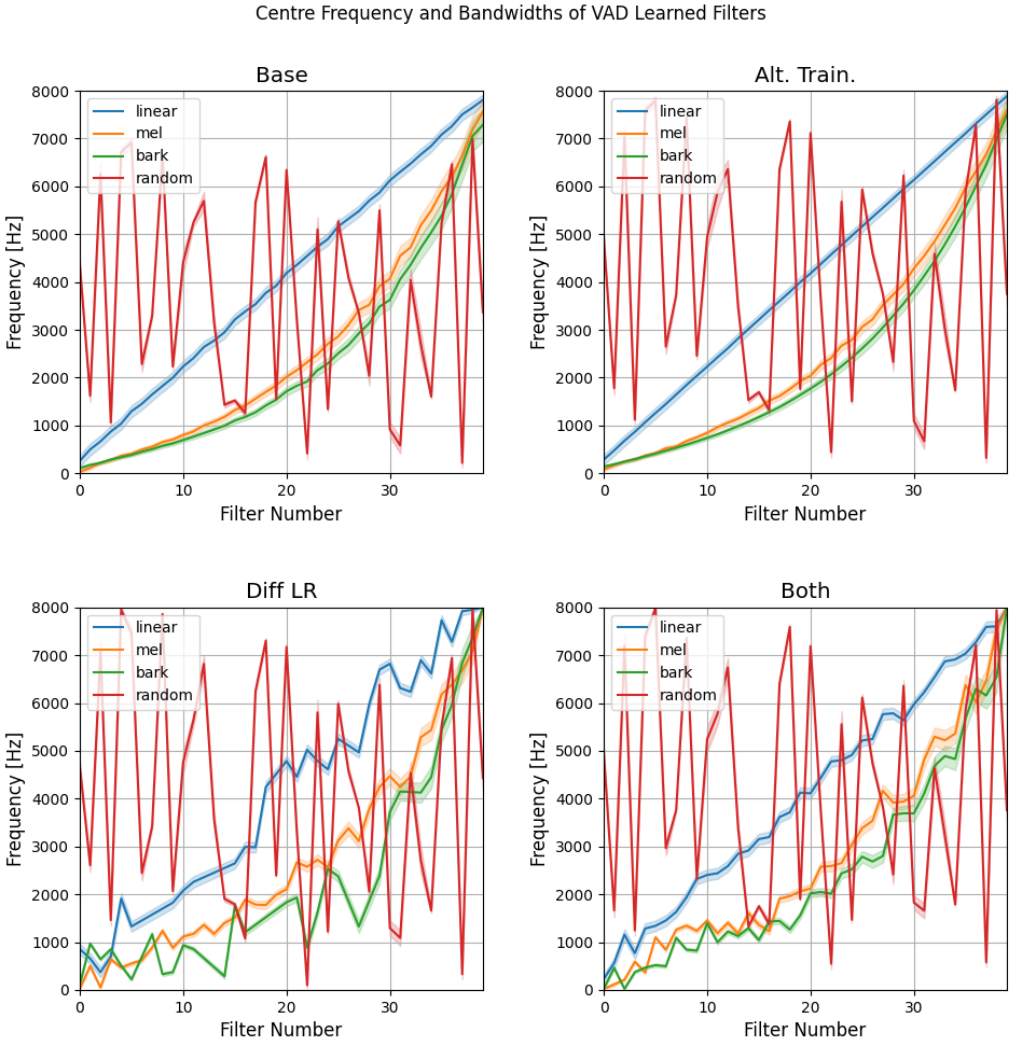

Figure 6.4: In each subplot in this figure, the frequency responses of the trained filterbanks for the VAD task are depicted. These subplots illustrate the frequency responses for each training strategy: *Baseline*, *Alt. Train*, *Diff LR* and *Both*. The solid line represents the center frequency, while the shaded area represents the bandwidth.

Similar to Section 5.4.3, the learned filters using the *Baseline* strategy exhibit little movement overall, with more pronounced shifts in filters with narrow initial bandwidth, notably in *Mel* and *Bark* initialisations. Again this is attributed to the small bandwidth and the nature of the JSD, which compares distributions. Slight changes in centre frequency lead to significant changes in distance. *Alt.Train.* similarly displays limited movement, allowing the same amount of movement over a shorter time period by restricting optimisation to only the frontend's parameters. Figure 6.4, reflects this lack of movement, with little change from the initial filters for both *Baseline* and *Alt.Train.* Despite some differences in the learned filterbanks between *Baseline* and *Alt.Train.*, the latter performs worse than Baseline on every initialisation in the VAD task.

Diff LR was shown in Table 6.1 to allow for the best and most consistent performance in the VAD task. The filters in this task also exhibit the most movement, as seen in Figure 6.3 and Figure 6.4. However, attention should be drawn to the training duration —- unlike models trained using *Baseline*, *Alt.Train.* or *Both* strategies, *Diff LR* creates an unstable training scenario resulting in training being interrupted (with the exception of *Mel-Diff LR* although this is likely by chance). The increased learning rate and aggressive optimisation lead to NaN values in the TF-representation which propagate through the model, causing an undefined loss and halting training. More robust parameter clamping and NaN handling in the frontend could address this issue. Nevertheless, *Diff LR* yields the best results on all initialisations, suggesting potential performance gains with modifications.

In the two psychoacoustic initialisations (*Mel* and *Bark*), most movement occurs in the low and middle frequency filters, consistent with *Baseline* and *Alt.Train.*, albeit on a larger scale. The speech energy in TIMIT is primarily located below 1 kHz, with little information past 3 kHz. These lower and middle frequency filters adjust during training to capture pertinent information close to their initialisation. Figure 6.4 highlights these changes from initialisation. However despite achieving similar results with all initialisations in Table 6.1, the learned filterbanks have not converged on a common arrangement.

The *Both* strategy seems to counteract the unstable training of *Diff LR*, allowing the model to train longer and while still showing movement in the filterbanks, as illustrated by both

Figure 6.3 and Figure 6.4. Despite stability and increased movement, models trained with *Both* perform worse than those using the less stable *Diff LR* or the simple *Baseline* strategies.

For the BSID task, Figure 6.5 illustrates the movement of filterbanks over time, while Figure 6.6 displays the frequency responses of each filterbank after training. Similar to Section 5.4.3, *Baseline* shows some movement, but not a radical change from the initialisation (see Figure 6.6). Referring to Figure 6.5, movement in the *Mel* and *Bark* initialisations is primarily observed in low and high frequencies, while the *Linear* initialisation shows small changes across all frequencies. The *Random* initialisation increases the bandwidth of some filters with minimal change to centre frequencies. This strategy provides the best overall performance on the task using the *Linear* initialisation (F1-Score 0.672, accuracy 0.729), and also provides the best performance using the *Bark* initialisation. However, as the filterbank undergoes minimal changes from its initialisation, it is believed that the performance improvements are primarily from the PCEN compression layer.

Alt.Train. provides more movement than *Baseline*, contrary to its behaviour in the VAD task. However, this often leads to decreased performance, with the exception of the *Mel* initialisation. The JSD (Figure 6.5) indicates significant movement in low-frequency filters and some in mid-frequency filters, particularly in the *Bark* initialised filterbank. Figure 6.6 illustrates that these changes result in higher centre frequencies and wider filter bandwidths, aligning with findings in Section 3.1.1 regarding the typical range of fundamental frequencies for bird vocalisations. While the learned *Mel* filterbank does become more 'bark-like', it is this filterbank that delivers the best performance. The *Linear* initialisation exhibits considerable movement across all frequencies, with slightly worse performance than *Baseline* (0.666 vs. 0.672). *Random* initialisation shows little movement and also poor performance when using the *Alt.Train.* strategy.

Unlike in the VAD task, *Diff LR* does not yield performance gains on any initialisation except for *Random*. As discussed earlier, the *Random* initialisation is not recommended and is solely used to test the mitigation strategies. On all other initialisations, performance decreases. *Diff LR* allows for substantial filter movement (Figure 6.5), although training is more stable on the BSID task. This is likely due to the higher fundamental frequency of bird

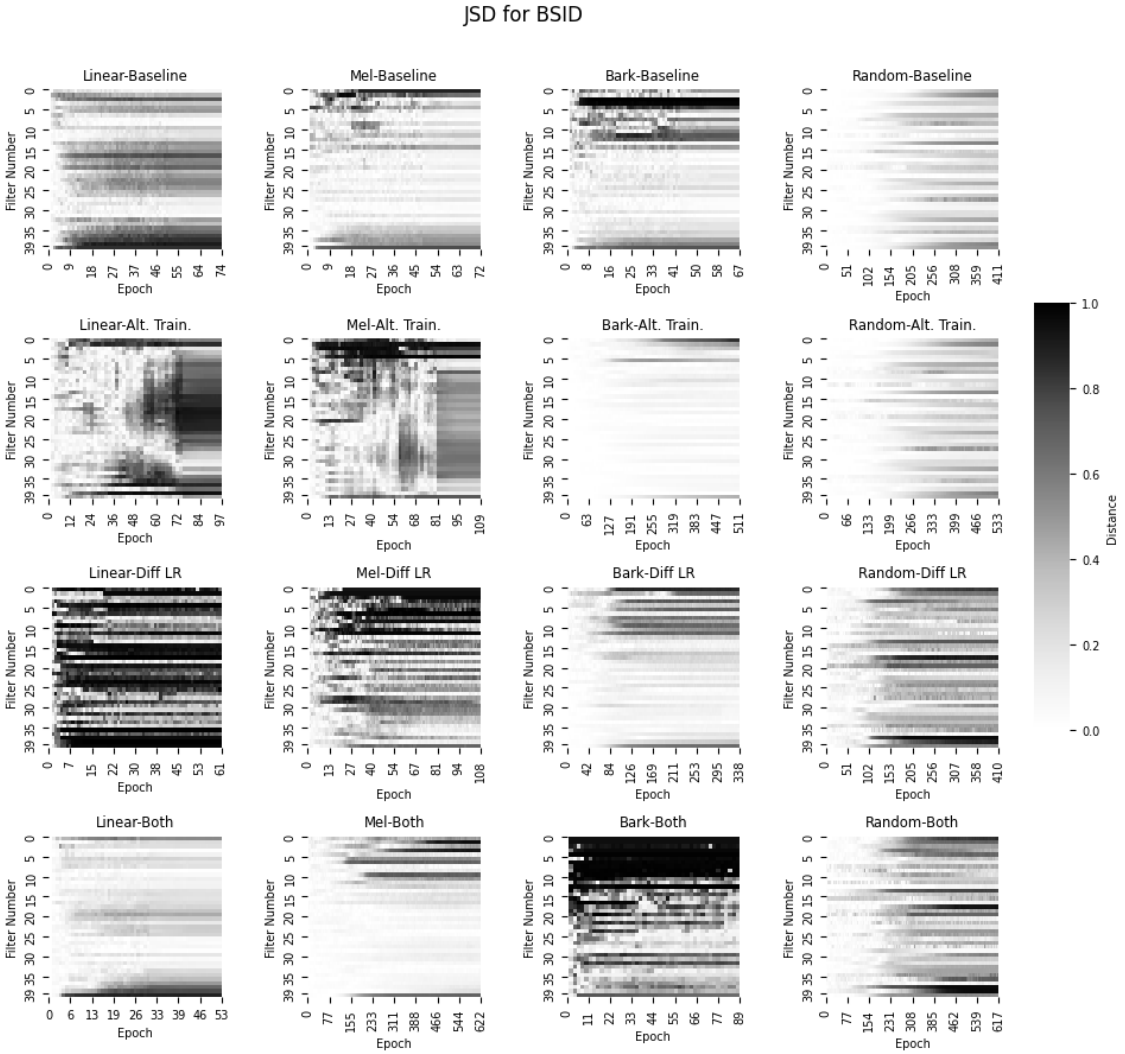

Figure 6.5: Jensen-Shannon distances of each filterbank from its initialisation for the BSID task. This figure is ordered by initialisation horizontally, and by training strategy vertically.

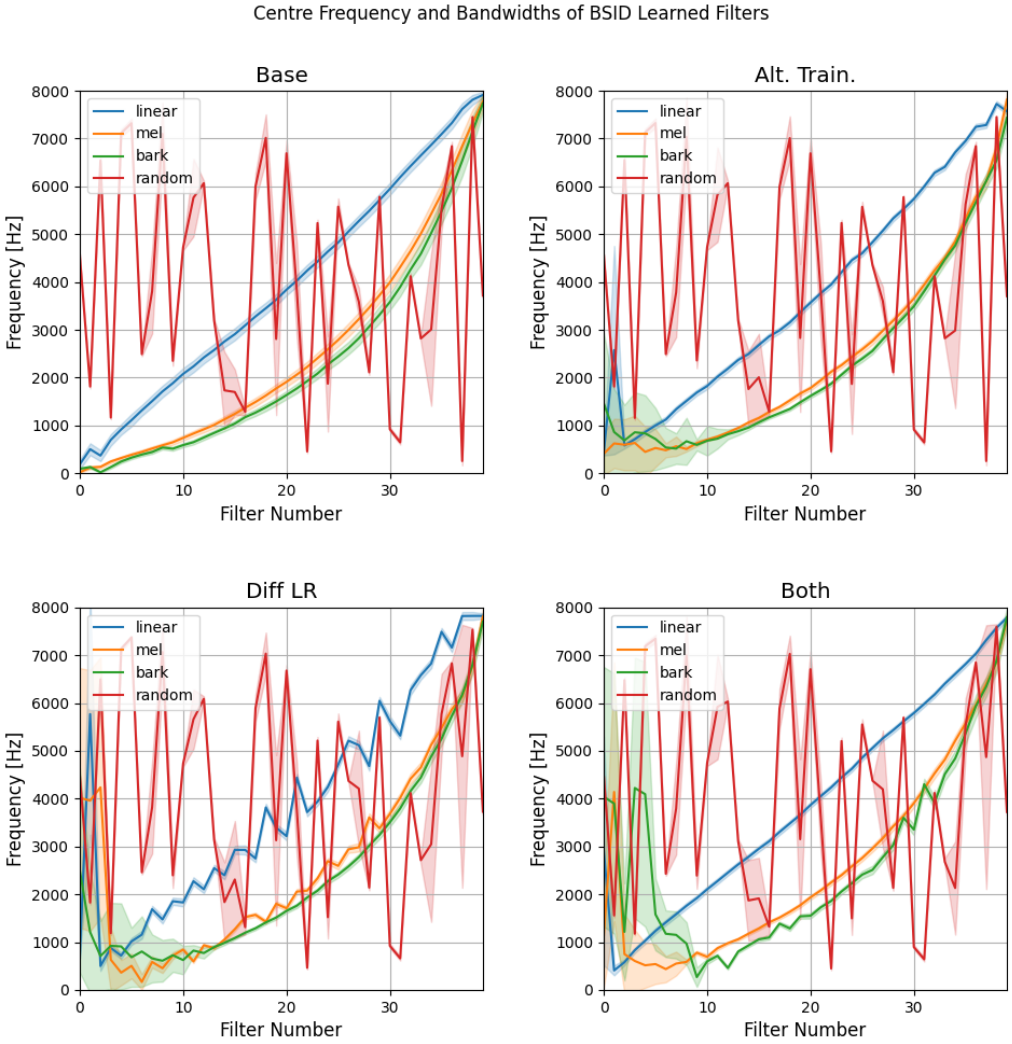

Figure 6.6: In each subplot in this figure, the frequency responses of the trained filterbanks for the BSID task are depicted. These subplots illustrate the frequency responses for each training strategy: *Baseline*, *Alt. Train*, *Diff LR* and *Both*. The solid line represents the center frequency, while the shaded area represents the bandwidth.

vocalisations, preventing the learned filters from going 'out-of-bounds'. The final learned filterbanks have similar characteristics to the *Alt.Train* filterbanks, except for the *Linear* initialisation, which exhibits more movement using this strategy.

The *Both* strategy generally performs poorly on all initialisations, except for *Bark* initialisation. Inspection of the resulting filterbank in Figure 6.6 suggests that wide bandwidth filters (in filters 0–8) covering a significant portion of the middle and high frequency ranges contribute to the relatively better performance. The learned *Mel* filterbank also shows similarities to its *Alt.Train.* and *Diff LR* counterparts, but achieves very poor performance.

The *Alt.Train.*, *Diff LR* and *Both* strategies encourage substantial movement in the low-frequency filters, particularly evident in the two psychoacoustic initialisations of *Mel* and *Bark*, as depicted in Figures 6.5 and 6.6. It is worth highlighting the average DFT of the BirdCLEF2021 dataset, illustrated in Figure 5.11, where a substantial amount of information is present between 2 kHz -– 5 kHz. In the psychoacoustic *Mel* and *Bark* initialisations the lower-frequency filters are initially positioned below 1 kHz, during training they expand towards higher centre frequencies and larger bandwidths in an effort to enhance their utility.

General Remarks

The overall poor performance of *Alt.Train.* (with the one exception in the BSID task) suggests that alternate training is an ineffective mitigation strategy for the filterbank initialisation problem and may even hinder performance. Additionally, *Both* also appears to be an ineffective mitigation strategy, primarily tempering the more aggressive optimisation of *Diff LR*. *Diff LR* serves as a promising starting point for formulating a mitigation strategy, delivering increased and consistent performance on the VAD task (the same is not seen in the BSID task). Additionally, the ease of implementation means that a general recommendation can be made to use separate optimisers.

Grouping frontend and backend parameters separately and applying the same optimisation algorithm and learning rate to both sets of parameters, is equivalent to employing one optimiser for all parameters in the model. This approach provides flexibility to optimise the frontend differently, utilising a distinct optimisation algorithm (with associated

hyperparameters), learning rate, scheduler, or a combination of these.

Concerning filterbank initialisation, it is unsurprising that a mel filterbank proves to be the best starting point for human speech, although a linear filter also performs well. Mel initialisation is appropriate owing to the large amount of filters below 1 kHz, effectively capturing a significant portion of the information present in human speech (refer to Figure 5.11). For bird vocalisations, linear initialisation appears to be the most effective. Linear initialisation does not prioritise frequencies below 1 kHz, where the frequency content of bird vocalisations rarely occurs (refer to Figure 5.11 and additionally Figure 3.2), offering greater flexibility when learning filterbanks. It also demonstrates good performance in the non-learnable case, as indicated in Table 5.6. The root cause of the filterbank initialisation problem, additional mitigation strategies, and the optimisation of filterbanks in general remain topics for future work. This is elaborated upon in Section 7.3 with possible lines of investigation discussed.

6.2 Returning to Few-Shot Learning

In the final paragraphs of Section 4.3.4, it was concluded that enhancing the performance of the FSL system discussed in Chapter 4 would be best achieved by focusing on improving feature representation. This decision was informed by an analysis of the embedding space, suggesting that the clustering of positive and negative class query points was influenced by noise and far-field conditions in the recordings. Improvements in feature representation would also have wider impact in tasks outside of an FSL context.

These improvements to feature representation involve the incorporation of learnable frontends, enabling the direct learning of features from the data. Chapter 5 explored various popular learnable frontends, some of which are proposed as 'off-the-shelf' replacements for the log-mel spectrogram. Section 5.3 evaluated these frontends in a bioacoustics task (species agnostic BAD), and found that learnable frontends do increase performance, particularly learnable compression such as PCEN [211]. While frontends employing learnable filterbanks such as LEAF [216] and TD [217] did not preform as well, their potential lies in the direct learning of filterbanks from the data, addressing the potential unsuitability of the

mel scale for bird audio.

After the investigations of Chapter 5 and the first part of the current chapter, we now revisit FSL. We use the same prototypical network (protonet) [177] architecture previously used in our submission [2] to the DCASE2021 Few-Shot Bioacoustic Sound Event Detection Challenge [125], with the addition of learnable frontends.

As this work focuses on feature representation and not improvements to the training regimen or model architecture, Section 6.2.1 provides a brief overview of advances in FSL for audio, particularly bioacoustics, since the conclusion of the 2021 challenge. Section 6.2.2 recaps the 2021 challenge task and data, and provides a description of the changes made to the protonet. Finally Section 6.2.3 presents and discusses the results of adding learnable frontends to the system.

6.2.1 Few-Shot Learning for Bioacoustics since 2021

As mentioned above, the protonet used in the following experiments was initially developed in early 2021 for the DCASE challenge. In the interest of direct comparison with other experiments presented in Chapter 4, and to focus solely on the impact of learnable frontends, this configuration remains unchanged. However, the DCASE Few-Shot Bioacoustic Sound Event Detection Challenge also occurred in 2022 and 2023, with advancements in the field with each edition.

In this section, an overview is provided of FSL in bioacoustics since 2021 to summarise advancements in the field parallel to this work. Due to changing datasets each year, reported metrics are not directly comparable across different years. Nonetheless, discussion will include each submission's ranking within its respective year to provide context of the submission's relative performance.

In the 2022 challenge, many entrants leveraged the methodology proposed by Yang et al. [214], employing transductive inference for addressing the FSL problem. The leading submission, presented by Tang et al. [191], achieved an F1-Score of 0.602 on that year's evaluation set. This submission noted the significance of support set construction to classification performance, emphasising that audio at event boundaries might poorly represent

the corresponding event. The author's hypothesis posited that audio from the middle of events serves as a better representation of the class. This approach involved using such audio for support set creation, employing adaptive window sizes during feature generation, and training the encoder with cross-entropy loss, similar to other non-meta-learning-based FSL systems [31, 195, 206]. Despite its effectiveness, this entry's model complexity was notably high, comprising 12M parameters, making it one of the largest models of that year.

The second-highest ranked system, submitted by Liu et al. [105], achieved an F1-Score of 0.482. This submission incorporated background class modelling, similar to our approach in Section 4.3.3, to better represent potential negative classes during inference. Additionally, it used transductive inference and a large pretrained CNN as the embedding function, requiring additional inference time training of the model and increasing model complexity.

The third place entry by Martinsson et al. [122] employed a protonet-based system trained with cross-entropy loss, achieving a F1-Score of 0.480. This submission used ensemble modelling and adaptive embedding functions based on the minimum length of a support set event. While the best results with this model involved ensemble modelling, pushing model complexity to 25 million parameters, the use of adaptive embedding functions alone yielded improvements over the baseline protonet model.

In 2023, the highest-ranking system was once again submitted by Tang et al. [213], with an F1-Score of 0.638. This was achieved through the use of a transformer-based architecture [201], capitalising on the temporal information of the audio, with a parameter count of approximately 21M parameters. The training process involved cross-entropy training, and akin to SimpleShot [206], the FSL problem was transformed into a fine-tuning problem using limited data, supplemented by data augmentation.

The second-highest ranked system in 2023, presented by Moummad et al. [126] achieved an F1-Score of 0.427. This entry utilised a supervised contrastive learning strategy [89], an extension of the triplet loss discussed in Section 4.3.2. Supervised contrastive learning aims to minimise the distance between similar instances while maximising the distance between dissimilar instances. Contrasting with triplet learning, which involves triplets of points (anchor, positive, and negative), supervised contrastive learning focuses solely on pairs of

points, whether similar or dissimilar. Triplet loss focuses on preserving relative distances between triplets, while supervised contrastive loss aims to learn discriminative representations by bringing similar samples close and separating dissimilar samples based on class labels.

The third place entry submitted by Liu et al. [106] achieved an F1-Score of 0.425. This protonet-based approach utilised a similar architecture to our system but incorporated Squeeze-and-Excitation (SE) blocks [74] to enhance performance. These SE blocks capture dependencies and global relationships between convolutional filters, generating scaling factors that manipulate and control the importance of each filter in the encoder network. This introduces an attention-like mechanism without significantly increasing computational complexity, maintaining a small model size of 724k parameters. While larger than our system's 132k parameters, SE blocks offer architectural performance improvements without a substantial increase in parameter count or relying on additional inference-time adaptation.

In summary, the top-performing systems in both 2022 and 2023 employ large architectures, yielding impressive results at the expense of complexity. Many approaches also incorporate transductive inference or other inference-time adaptations to the model. These approaches are computationally intensive and potentially unsuitable for deployment on low-resource hardware.

Nonetheless, there are other techniques from these entries that do not rely on large models and inference-time adaptation. Transitioning from prototypical loss functions to supervised contrastive learning [106] or cross-entropy can improve performance. The inclusion of SE blocks [106] represents an architectural improvement with relatively low increased computational complexity, especially when compared to other entries. Improved support set construction [191] contributes to the development of more robust class prototypes. The application of these methods to a low-resource system, while also incorporating learnable frontends, is suggested as future work.

6.2.2 Experimental Setup

The experimental setup closely mirrors that in Section 4.2, although a brief overview of the task, training data and evaluation data is provided here for convenience. The DCASE

challenge [125] entails an activity detection task, wherein each recording contains a single class of interest, either a mammal or bird vocalisation. The objective is to accurately identify onset and offset boundaries for events corresponding to the class of interest given the positive and negative support sets. The model's support set consists of the first five instances of the class of interest in each recording and the model is required to generalise to classes unseen during training time. Training involves multiclass training, while inference operates as a binary classifier predicting the presence of the class of interest in an audio segment.

For detailed information on training and evaluation data, refer to Section 4.2.2, and the summaries in Table 4.1 and Table 4.3. The training set annotations are multiclass, containing the audio filename, onset and offset times, and positive (POS), negative (NEG), and unknown (UNK) values for each class. Evaluation set annotations are binary with a skewed class distribution, primarily negative. Support sets are constructed from the first five positive events and the preceding audio segments for the negative class. As these audio events are of varying duration, the first 5 entries for both classes will result in a different amount of support audio.

The architecture of the prototypical network remains unchanged from Chapter 4 (see Table 4.4), except for the addition of a learnable frontend before the encoder. The primary changes pertain to data loading: the raw waveform is now segmented into 200 ms events with a 50 ms hop, eliminating the use of precomputed mel spectrograms and static PCEN compression. No data augmentation is performed, but Random Over-Sampling is still employed to address the issue of class imbalance.

Two learnable frontends, a fully trainable PCEN compression layer [211] and eLEAF [166, 216] (with PCEN), are chosen based on their performance in Section 5.4. PCEN expects a TF-representation, computed from the raw waveform's STFT with 128 frequency bins. No frequency warping is performed, based on the results from Section 5.4.3 and Section 6.1.4 where a linear frequency scale yielded the best performance. eLEAF is configured similarly, with 128 filters and linear initialisation. Both frontends produce a TF-representation of $\mathbb{R}^{17\times128}$, the same size as the original TF-representations. No strategy from Section 6.1 is utilised during training of the frontends, as this task closely resembles the BSID task, which

System	F1-Score	Precision	Recall
Original Submission [2]	0.350	0.488	0.272
Modified System (Section 4.3)	0.368	$\mathbf{0.522}$	0.284
PCEN	$\mathbf{0.438}$	0.497	$\mathbf{0.392}$
eLEAF	0.430	0.481	0.389
2021 Challenge Winner [214]	0.384	N/A	N/A

Table 6.2: Results on the challenge evaluation set for the original submitted system, employing prototypical triplet loss + multiple representations (Modified System), and when utilizing learnable frontends (PCEN and eLEAF). The F1-Score of the top-ranked system [214] from the 2021 challenge is included for comparison, precision and recall values are unavailable. The best results for each metric are marked in **bold**.

performed best using no mitigation strategy.

Training employs the prototypical triplet loss [48], detailed in Section 4.3.2, and follows the same procedure outlined in Section 4.2.3 with SGD optimisation, an initial learning rate of 0.01, and a momentum factor of 0.85. Learning rate is scheduled to halve when a plateau is reached, with a patience of 5 epochs and a threshold of 0.01.

6.2.3 Results & Discussion

Table 6.2 presents a comparative analysis of the performance of our original protonet system, the system incorporating modifications outlined in Section 4.3, and protonets featuring learnable frontends (PCEN and eLEAF). For reference, the performance of the top-ranking entry [214] from the 2021 challenge is also included for comparison, although no precision and recall values were provided. Table 6.2 can be cross-referenced with Table 4.7 and Table 4.8, which also report on the evaluation set, and entries relevant to this discussion have been included for ease of comparison.

Comparing the performance across all systems, it is evident that learnable frontends substantially improve the the prototypical network's performance. The use of either PCEN or eLEAF yields better performance in both precision and recall compared to both the system with modifications from Section 4.3 and the top ranking system from the 2021 challenge. The 2021 top-ranking system by Yang et al. [214], is a transductive inference based system, requiring additional training steps during inference time and requiring 468k parameters. In

contrast, our prototypical network-based system requires no extra training and has a maximum of 133k parameters (using 128 filters, PCEN adds 512 parameters, while eLEAF adds 896).

Using a PCEN frontend yields a relative improvement of 25% in F1-Score, 2% in precision and 44% in recall. On the other hand, an eLEAF frontend offers a relative improvement of 22% in F1-Score, 43% in recall, but a relative deterioration of 1.5% in precision. Notably, both learnable frontends exhibit decreased precision compared to the modified system. Overall the adoption of learnable frontends has substantially increased recall with a marginal sacrifice in precision, resulting in an overall higher F1-Score. This means a reduction in false negatives, the prevalence of which was attributed to noisy and far-field audio in Section 4.3.4, motivating the use of learnable frontends. However, this comes at the expense of increased false positives, which will be further discussed in the subsequent analysis of the embedding space for both the PCEN and eLEAF networks.

Between PCEN and eLEAF, PCEN outperforms in all three metrics, aligning with earlier findings in this work (Section 5.3.4), where PCEN exhibited superior performance in a BAD task that also included LEAF. The performance gap between PCEN and eLEAF has seemingly decreased, which may be due to the more aggressive optimisation of the protonets compared to the EfficientNet-based model from Section 5.3. However, a direct comparison is challenging due to differences in model and paradigm between these instances and those discussed in Section 5.3.4.

Examining the impact of learnable frontends involves an analysis of the embeddings generated by the network. As in Chapter 4, t-SNE is employed to reduce the dimensionality of the embedding space, utilising the same hyperparameters as in Section 4.3.1. Figures 6.7 and 6.8 depict t-SNE visualisations of protonets using PCEN and eLEAF, respectively. As the original challenge submission and the modified system are also included for comparison in Table 6.2, this discussion will also refer to Figures 4.3 and 4.6, which show the t-SNE visualisations of the embedding spaces for the original challenge submission and the modified system.

When comparing all four visualisations, the overall structure is not radically changed. This is expected. Despite the improved performance with learnable frontends, the network still

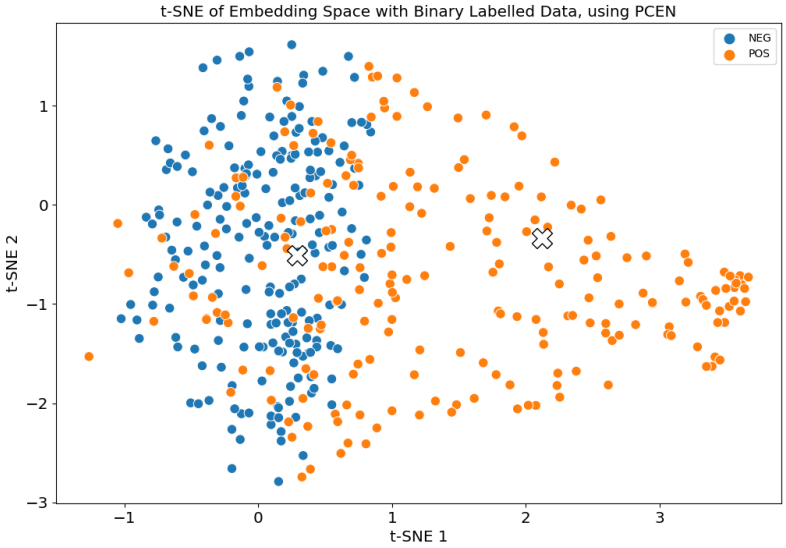

Figure 6.7: t-SNE projection of the test data embeddings learned by the prototypical network trained using PCEN as the frontend. The prototype representations for both positive and negative classes are marked by a white 'X'.

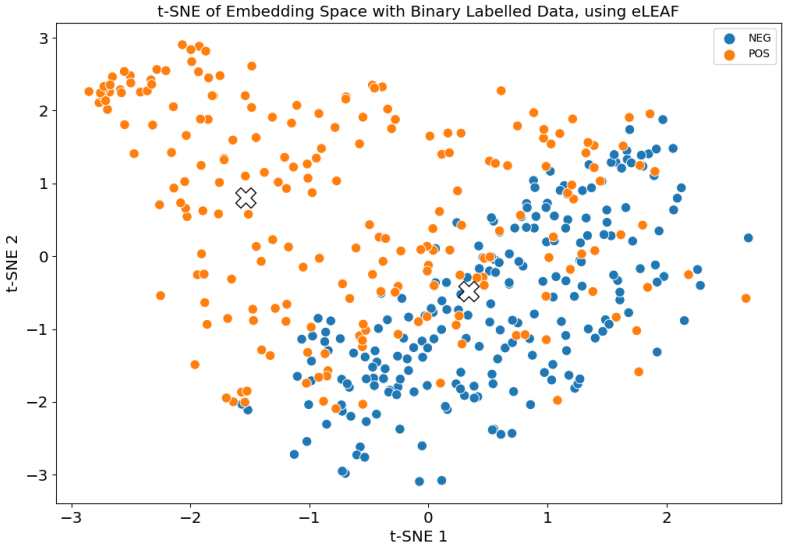

Figure 6.8: t-SNE projection of the test data embeddings learned by the prototypical network trained using eLEAF as the frontend. The prototype representations for both positive and negative classes are marked by a white 'X'.

exhibits type 1 (false positive) and type 2 (false negative) errors due to insufficient clustering and separation between classes. However, both networks with learnable frontends enhance the recall metric, reducing the number of false negatives. In Figures 4.3 and 4.6, these false negatives are clearly seen, with many positive query points near the negative prototype and within the negative cluster. In contrast, Figures 6.7 and 6.8 show a reduction in these false negatives near the negative prototype, contributing to the increased recall and overall performance improvement. The primary contributor to this enhanced performance is suspected to be the PCEN compression, leading to similar overall structures in the embedding spaces of models using PCEN and eLEAF.

False positives remain an issue, and learnable frontends have not improved on the precision metric. These false positives can be attributed to non-stationary background noise, sound events related to but not belonging to the class of interest, and insufficient modelling of background classes. Although there is some deterioration of the precision metric compared to the modified protonet from Section 4.3, it is not substantial, especially considering the gain in recall. Further research may be needed to model the background class, enhance feature representations further, or develop more discriminative embedding spaces, perhaps taking inspiration from ideas discussed in Section 6.2.1.

Both frontends demonstrate comparable performance, and the majority of this performance is attributed to the introduction of a fully trainable PCEN layer, with the learnable filterbank having minimal impact. Learnable compression of the spectrogram magnitudes using PCEN contributes the most to performance improvements, a trend observed in the wider evaluation of learnable frontends in Section 5.3. Despite eLEAF's improved throughput compared to the original implementation, generating a TF-representation with eLEAF's filterbank is slower than calculating the STFT of the same audio signal. Coupled with the reduced performance compared to PCEN, suggests that PCEN is the more suitable learnable frontend for this task.

These experiments show that learnable frontends also improve performance in an FSL setting, as they do in a conventional classification setting (Section 5.3.4). Concurrently, the field of FSL for bioacoustics has progressed, and additional techniques are available to improve performance further. As mentioned earlier in Section 3.4, while FSL is attractive for

bioacoustics due to the limited availability of fully annotated datasets, its capacity to generalise to unseen classes and potential computational efficiency are also attractive. Some systems in Section 6.2.1 rely on methods requiring additional inference-time training or network architectures unsuitable for low-resource deployment. However, there are applicable methods and techniques, such as those discussed in the final paragraph of Section 6.2.1.

The further application of learnable frontends to newer, more effective architectures and techniques, while continuing to consider low-resource applications, is left for future work. This, along with other future work specific to learnable frontends mentioned in Section 6.1.4, constitutes the discussion in Section 7.3.

Chapter 7

Conclusion

This thesis presents features and models designed to advance the automatic and remote monitoring of birds based on their vocalisations, with an emphasis on prioritising potential deployment to low-resource hardware. Additionally, it explores the efficacy of learnable frontends in bioacoustic tasks and addresses a prevalent issue identified in learnable filterbank literature. With the anticipated changes in climate and environment in the coming decades, the long-term monitoring of bird populations has become increasingly vital for scientific research, conservation efforts, and ecological considerations. Manual monitoring, being labour-intensive, poses a challenge as the time spent collecting data could be better utilised for analysis. Therefore, the development of high-performing, efficient systems capable of operating on low-resource devices suitable for field deployment is crucial.

This chapter summarises the contributions and findings of each section (explicitly laid out in Section 1.4), aligning with the two research questions proposed in the introduction (Section 1.2). These research questions focus on the development of low-resource automatic monitoring systems and the applicability of learnable frontends to bird audio and bioacoustics. Additionally, this chapter discusses potential avenues for future work on FSL for bioacoustics and for learnable frontends more broadly, as well as some final remarks on this thesis.

7.1 Architectures for Monitoring Bird Populations with Low Resource Hardware

To address **RQ1** and its two sub-questions, **RQ1.1** and **RQ1.2**, Chapters 3 and 4 present approaches for species-agnostic activity detection in bird monitoring. This is a vital first step in the monitoring process and can function as a pre-filtering step on low-resource hardware to save storage or minimise the volume of data to be analysed off-site.

Chapter 3 delves into addressing **RQ1.1** by developing an original approach to species-agnostic BAD using low-resource classifiers such as random forests, alongside AM, spectral and pitch-based features. This feature set, named AMPS, capitalises on an understanding of the relationship between a bird's ability to modulate amplitude and frequency. AMPS, with random forests, exhibits promising results as an initial filtering step, achieving a balance between accuracy on bird audio tasks and computational efficiency. AMPS is the primary contribution from Chapter 3. The algorithms used are within the capabilities of embedded devices, offering performance only slightly below that of a small CNN, at a fraction of the computational cost at inference time. Performance using AMPS also surpasses that of other features such as MFCCs.

However, despite the promising aspects of the AMPS feature set, which is more grounded in signal processing than machine learning, there are issues related to feature corruption due to noise, similar to MFCCs. Consequently, low SNR recordings are prone to misclassification, and the model's limited generalisation to unseen species in the test set poses issues. While suitable for deployment on low-resource devices, the potential issues with feature corruption and generalisation mean that AMPS with random forests do not definitively address **RQ1**.

As large, temporally detailed, fully annotated bird audio datasets necessary for generalisation are limited, the emergence of FSL for bioacoustics prompted the inclusion of **RQ1.2**. Chapter 4 presents a protonet FSL system submitted to the DCASE2021 challenge and analyses the system through an examination of the embedding space. This analysis identified key areas for further development, namely background class modelling and improving both inter-cluster separation and intra-cluster cohesion of the embeddings. Neither of these

strategies improved performance of the system substantially, with further analysis suggesting that research effort would be better spent on improving feature representations by learning from the data itself. This motivates the exploration of **RQ2**, regarding the use of learnable frontends in bioacoustics.

In Chapter 6, FSL is revisited to apply findings from investigations into learnable frontends and their applicability to bioacoustics. The proposed system is the first to successfully incorporate fully learnable frontends within the few-shot learning model. The experiments in this chapter conclusively demonstrate that learnable frontends significantly enhance performance, outperforming the top submission from the 2021 challenge. Although the results are favourable given the class imbalance and the field's relative immaturity, we acknowledge the need for further research to enhance the system's performance. Further application of learnable frontends to FSL and the development of more reliable FSL-based activity detection are areas for future research and we believe FSL has continued promise in bioacoustic activity detection more generally.

7.2 Learnable Frontends in Audio Deep Learning

To address **RQ2** and its two sub-questions, **RQ2.1** and **RQ2.2**, chapters 5 and 6 provide insights into the use and characteristics of learnable frontends. This includes an evaluation of various learnable frontends in the context of a shared bird audio task, a detailed examination of the shortcomings of learnable filterbank-based frontends — specifically, the filterbank initialisation problem — and the introduction of training-based strategies to mitigate this issue. The chapters also offer general recommendations on the use of learnable frontends in bioacoustics.

In Chapter 5, a benchmark of traditional and 'off-the-shelf' learnable acoustic frontends is conducted for audio classification, focusing on a bioacoustics task, specifically species-agnostic BAD. This is the most comprehensive investigation to date on the suitability of learnable frontends in bird audio. The results demonstrate significant improvements in model accuracy using learnable frontends compared to traditional STFT or mel spectrogram features. PCEN emerges as the top performer overall, a finding corroborated in Chapter 6.

The benchmarking and application of learnable frontends address **RQ2.1**, providing insights into their impact on bird audio tasks and concluding that PCEN with a linear scale TF-representation is recommended for optimal performance in terms of metrics and computation.

The latter section of Chapter 5 characterises the filterbank initialisation problem, addressing **RQ2.2**. To justify the use of learnable filterbanks, they should yield reliable performance increases. Ideally, learnable filterbanks should move from their initialisation to a set of optimal filters. The methodology and results in Section 5.4 provide novel insights into quantifying the shortcomings of learnable filterbank-based frontends. The sensitivity of learnable filterbanks to initialisation is demonstrated, and the change from initialisation to final learned filters is quantified through the JSD and analysis of frequency responses. The lack of learning in learnable filterbanks was consistent across two different tasks and four initialisation strategies and while performance was improved with learnable filterbanks, there was additional computational cost. This conclusively answers **RQ2.2**, indicating that learnable frontends are highly sensitive to their initialisation, but the underlying reasons remain a subject for future work.

The inconsistency in the performance of the learned filters, coupled with the lack of movement from initial filterbank values, demonstrates a shortcoming in the overall optimisation strategy. Attempts to address these shortcomings in Chapter 6, derived from the conclusions arising from **RQ2.2**, yielded mixed results, with some strategies performing consistently well in one task but not the other. This emphasises the need for further research into the root causes of the filterbank initialisation problem to develop effective mitigation strategies. Nevertheless, it is recommended to use separate optimisers for frontend and backend parameters for flexibility, and a higher frontend learning rate promotes filterbank movement. While learnable filterbanks show promise and merit further research, they cannot currently be recommended for effective, practical systems. However, PCEN compression yields impressive results, and is the recommended choice for bioacoustics, particularly in bird audio applications.

7.3 Future Work

The results, discussions, conclusions and limitations of this work, alongside developments concurrent to this work, suggest a number of avenues for future work. Three of these areas of future work are discussed here.

7.3.1 Few-Shot Learning for Bioacoustics

FSL for bioacoustics is still a young field with considerable potential, applicable to scenarios involving the availability of compute power, to low-resource applications. Each iteration of the DCASE FSL challenge shows improvement with more diverse data and an increasing need for generalisation. However, the performance of these systems still falls short and necessitates further refinement before practical deployment in real-world settings.

The incorporation of recent techniques discussed in Section 6.2.1, in conjunction with learnable frontends, which are shown to increase performance (Section 6.2.3), would be a good starting point for future work. While approaches like transductive inference or other methods requiring extensive inference-time adaptation may offer significant performance boosts over metric-learning methods like protonet, they should be avoided if the objective is real-time classification and deployment on low-resource hardware. Similarly, the utilisation of large systems with millions of parameters, despite the potential for pruning and quantisation, should be avoided.

In addition to advancing FSL, realising the full potential of FSL as a low-resource activity detector requires the implementation of such networks on embedded hardware. This step is crucial for ensuring practical deployment in scenarios where computational resources are limited. In general, the application of deep learning systems for bioacoustics to low-resource hardware is an under-researched area in the field.

7.3.2 Causes of the Filterbank Initialisation Problem and Mitigation Strategies

The underlying causes of the learnable filterbank initialisation problem remain undiscovered. Due to time constraints, this aspect was not extensively researched or presented in this

thesis. A more comprehensive understanding of the root causes of this problem would serve as the foundation for more effective mitigation strategies for existing learnable filterbanks or the design of a frontend that addresses this issue.

As discussed above the primary focus, with regards to learnable filterbanks, in this work has been on time-domain filterbanks and the flexibility provided by the scattering transform in transforming hyperparameters into trainable parameters. Specifically, the attention has been on parameterised time-domain filterbanks such as LEAF. Neural networks optimised through gradient-based methods require the calculation of gradients for each layer, and in a parameterised time-domain filterbank, these gradients exhibit a sinusoidal component. This sinusoidal component makes the filterbank more susceptible to getting trapped in local minima. This is actually a strike against parameterised time-domain filterbanks specifically, as unconstrained time-domain filterbanks such as TD do not possess sinusoidal gradients, though they still share sensitivity to initialisation (albeit to a lesser extent). Future research should explore the impact of different optimisation algorithms on learnable filterbanks and conduct a more thorough investigation of their gradients.

7.3.3 Learnable Frontends

More generally, learnable frontends are an area with huge potential for further development. Learnable compression, exemplified by effective techniques like PCEN, is a well established approach that is gaining increased adoption. While initially only one aspect of PCEN was trainable, incremental improvements have now enabled the training of all parameters. However, as pointed out by Schlüter and Gutenbrunner [166], PCEN is not suited for parallel hardware, due to the smoother component, which is implemented as an IIR filter. Further development of compression layers that provide similar effects on the time-frequency representation as PCEN may be beneficial, although it may not be the most rewarding direction for future work, as PCEN is not a major bottleneck.

The mitigation strategies proposed in Section 6.1 were derived empirically and based on insights gained while characterising the filterbank initialisation problem in Section 5.4. However, these strategies lack insight from a rigorous investigation into the root causes.

Among the proposed strategies, only *Diff LR* consistently improved performance, and even then, it was effective on only one task/architecture while introducing stability issues. While a recommendation is made to incorporate *Diff LR* into models employing learnable frontends, it is acknowledged that more suitable mitigation strategies or entirely new learnable filterbank architectures will emerge from a thorough investigation into the root causes of the initialisation problem.

A suggestion is to make use of a hybrid approach, utilising the frequency-domain representation of the filterbank as part of training. This idea draws inspiration from Peng et al.'s [143] work, where a sparsity constraint on their frequency-domain learnable filterbank encouraged sensible movement of the filters. In their work variations in center frequency, bandwidth, and filter density were observed and the resulting filters remained interpretable due to the sparsity constraints. This approach could be applied to the creation of time-domain filterbanks by using the center frequency and bandwidth from these filters as parameters for a parameterised time-domain filterbank. This approach would circumvent the need to employ the STFT in feature creation. Instead, it would harness the frequency domain representations of filters to generate sensible data-driven filters, different from their initialisation.

7.4 Final Remarks

It is hoped that the findings presented in this thesis, along with the associated publications, will serve to encourage further research in two areas of audio processing. Firstly, the further development of high-performing, efficient monitoring systems capable of running on constrained devices suitable for remote deployment is crucial for conservation. While many approaches demonstrate impressive results by utilising more data and larger models, these models are complex and unsuitable for large-scale deployment. Automatically monitoring potentially rare and at-risk populations in remote locations will only occur if the tools available to ornithologists are easily deployed, readily available and match their hardware budget. Additionally it is hoped that readers will approach learnable filterbanks in a more analytical way, aiming to design filterbanks that can generate an optimal set of filters for a

given task and domain. In numerous tasks, including bioacoustics, the mel scale may not be ideal or suitable, and the hope is that through learning directly from the data related to a specific task, suitable filterbanks can be learned.

Bibliography

[1] Joakim Andén and Stéphane Mallat. "Deep Scattering Spectrum". In: *IEEE Transactions on Signal Processing* 62.16 (2014), pp. 4114–4128.

[2] Mark Anderson and Naomi Harte. *Bioacoustic Event Detection with Prototypical Networks and Data Augmentation*. Tech. rep. DCASE2021 Challenge, June 2021.

[3] Mark Anderson and Naomi Harte. "Learnable Acoustic Frontends in Bird Activity Detection". In: *2022 International Workshop on Acoustic Signal Enhancement (IWAENC)*. 2022, pp. 1–5. DOI: 10.1109/IWAENC53105.2022.9914694.

[4] Mark Anderson, John Kennedy, and Naomi Harte. "Low Resource Species Agnostic Bird Activity Detection". In: *2021 IEEE Workshop on Signal Processing Systems (SiPS)*. 2021, pp. 34–39. DOI: 10.1109/SiPS52927.2021.00015.

[5] Mark Anderson, Tomi Kinnunen, and Naomi Harte. "Learnable Frontends That Do Not Learn: Quantifying Sensitivity To Filterbank Initialisation". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095474.

[6] Sercan O. Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. *Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting*. 2017. arXiv: 1703.05390 [cs.CL].

[7] Randall Balestriero, Romain Cosentino, Hervé Glotin, and Richard Baraniuk. "Spline Filters for End-to-End Deep Learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 364–373.

[8] A. Banerjee, Xin Guo, and Hui Wang. "On the Optimality of Conditional Expectation as a Bregman Predictor". In: *IEEE Transactions on Information Theory* 51.7 (2005), pp. 2664–2669. DOI: 10.1109/TIT.2005.850145.

[9] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. "Clustering with Bregman Divergences". In: *Journal of Machine Learning Research* 6.58 (2005), pp. 1705–1749. URL: http://jmlr.org/papers/v6/banerjee05b.html.

[10] G. J. L. Beckers, R. A. Suthers, and C. Ten Cate. "Pure-tone Birdsong by Resonance Filtering of Harmonic Overtones". In: *Proceedings of the National Academy of Sciences of the United States of America* 100.12 (2003), pp. 7372–7376.

[11] G.J.L. Beckers. "Bird Speech Perception and Vocal Production: A Comparison with Humans". In: *Human Biology* 83.2 (2011), pp. 191–212. DOI: 10.3378/027.083.0204.

[12] James C Bezdek and Richard J Hathaway. "Convergence of Alternating Optimization". In: *Neural, Parallel & Scientific Computations* 11.4 (2003), pp. 351–368.

[13] James C Bezdek and Richard J Hathaway. "Some Notes on Alternating Optimization". In: *Advances in Soft Computing—AFSS 2002: 2002 AFSS International Conference on Fuzzy Systems Calcutta, India, February 3–6, 2002 Proceedings*. Springer. 2002, pp. 288–300.

[14] Radoslaw Bielecki. *Few-Shot Bioacoustic Event Detection with Prototypical Networks, Knowledge Distillation, and Attention Transfer Loss*. Tech. rep. DCASE2021 Challenge, June 2021.

[15] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. "Choosing Starting Values for the EM Algorithm for getting the Highest Likelihood in Multivariate Gaussian Mixture Models". In: *Computational Statistics & Data Analysis* 41.3 (2003). Recent Developments in Mixture Model, pp. 561–575. DOI: https://doi.org/10.1016/S0167-9473(02)00163-9.

[16] Malik Boudiaf, Hoel Kervadec, Ziko Imtiaz Masud, Pablo Piantanida, Ismail Ben Ayed, and Jose Dolz. *Few-Shot Segmentation Without Meta-Learning: A Good Transductive Inference is All You Need?* 2021. arXiv: 2012.06166 [cs.CV].

[17] Malik Boudiaf, Imtiaz Ziko, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. "Information Maximization for Few-shot Learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2445–2457.

[18] L.M. Bregman. "The Relaxation Method of Finding the Common Point of Convex Sets and its Application to the Solution of Problems in Convex Programming". In: *USSR Computational Mathematics and Mathematical Physics* 7.3 (1967), pp. 200–217. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(67)90040-7.

[19] Judith C. Brown. "Calculation of a Constant Q Spectral Transform". In: *The Journal of the Acoustical Society of America* 89.1 (Jan. 1991), pp. 425–434. DOI: 10.1121/1.400476.

[20] H. Brumm. "The Impact of Environmental Noise on Song Amplitude in a Territorial Bird". In: *Journal of Animal Ecology* 73.3 (2004), pp. 434–440.

[21] H. Brumm and A. Zollinger. "The Evolution of the Lombard effect: 100 years of Psychoacoustic Research". In: *Behaviour* 148.11-13 (2011), pp. 1173–1198.

[22] T Tony Cai and Rong Ma. "Theoretical Foundations of t-SNE for Visualizing High-dimensional Clustered Data". In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 13581–13634.

[23] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen. "Convolutional Recurrent Neural Networks for Bird Audio Detection". In: *25th European Signal Processing Conference, EUSIPCO 2017* 2017-January (2017), pp. 1744–1748. DOI: 10.23919/EUSIPCO.2017.8081508.

[24] Emre Cakir, Ezgi Can Ozan, and Tuomas Virtanen. "Filterbank Learning for Deep Neural Network Based Polyphonic Sound Event Detection". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, pp. 3399–3406. DOI: 10.1109/IJCNN.2016.7727634.

[25] C. K. Catchpole. "Production and Perception". In: *Bird Song: Biological Themes and Variations, Second Edition*. 2008, pp. 19–48.

[26] C. K. Catchpole. "The Study of Birdsong". In: *Bird Song: Biological Themes and Variations, Second Edition*. 2008, pp. 1–18.

[27] C. K. Catchpole. "Themes and Variations". In: *Bird Song: Biological Themes and Variations, Second Edition*. 2008, pp. 203–234.

[28] Yair Censor and Arnold Lent. "An Iterative Row-action Method for Interval Convex Programming". In: *Journal of Optimization Theory and Applications* 34 (1981), pp. 321–353.

[29] F Stuart Chapin III, Erika S Zavaleta, Valerie T Eviner, Rosamond L Naylor, Peter M Vitousek, Heather L Reynolds, David U Hooper, Sandra Lavorel, Osvaldo E Sala, Sarah E Hobbie, et al. "Consequences of Changing Biodiversity". In: *Nature* 405.6783 (2000), pp. 234–242.

[30] Da Chen, Yuefeng Chen, Yuhong Li, Feng Mao, Yuan He, and Hui Xue. "Self-Supervised Learning for Few-shot Image Classification". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 1745–1749.

[31] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. "A Closer Look at Few-shot Classification". In: *arXiv preprint arXiv:1904.04232* (2019).

[32] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. "Meta-baseline: Exploring Simple Meta-learning for Few-shot Learning". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9062–9071.

[33] Hao Cheng, Chenguang Hu, and Miao Liu. *Prototypical Network for Bioacoustic Event Detection via i-Vectors*. Tech. rep. DCASE2021 Challenge, June 2021.

[34] Kin Wai Cheuk, Hans Anderson, Kat Agres, and Dorien Herremans. "NNaudio: An On-the-fly GPU Audio to Spectrogram Conversion Toolbox using 1D Convolutional Neural Networks". In: *IEEE Access* 8 (2020), pp. 161981–162003.

[35] Wei Chu and Abeer Alwan. "FBEM: A Filter Bank EM algorithm for the Joint Optimization of Features and Acoustic Model Parameters in Bird Call Classification". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2012, pp. 1993–1996. DOI: 10.1109/ICASSP.2012.6288298.

[36] James W Cooley and John W Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series". In: *Mathematics of computation* 19.90 (1965), pp. 297–301.

[37] Aurora Linh Cramer, Vincent Lostanlen, Andrew Farnsworth, Justin Salamon, and Juan Pablo Bello. "Chirping up the Right Tree: Incorporating Biological Taxonomies into Deep Bioacoustic Classifiers". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 901–905.

[38] Wentao Cui and Yuhong Guo. "Parameterless Transductive Feature Re-representation for Few-shot Learning". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 2212–2221.

[39] Kevin Darras, Péter Batáry, Brett J. Furnas, Ingo Grass, Yeni A. Mulyani, and Teja Tscharntke. "Autonomous Sound Recording Outperforms Human Observation for Sampling Birds: A Systematic Map and User Guide". In: *Ecological Applications* 29.6 (2019), e01954. DOI: https://doi.org/10.1002/eap.1954.

[40] S. Davis and P. Mermelstein. "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4 (1980), pp. 357–366. DOI: 10.1109/TASSP.1980.1163420.

[41] A. De Cheveigné. "YIN, a Fundamental Frequency Estimator for Speech and Music". In: *Journal of the Acoustical Society of America* 111.4 (2002), pp. 1917–1930. DOI: 10.1121/1.1458024.

[42] Tom Denton, Scott Wisdom, and John R Hershey. "Improving Bird Classification with Unsupervised Sound Separation". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 636–640.

[43] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification". In: *Proc. Interspeech 2020*. 2020, pp. 3830–3834. DOI: 10.21437/Interspeech.2020-2650.

[44] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. "A Baseline for Few-shot Image Classification". In: *arXiv preprint arXiv:1909.02729* (2019).

[45] A. Digby, M. Towsey, B. D. Bell, and P. D. Teal. "A Practical Comparison of Manual and Autonomous Methods for Acoustic Monitoring". In: *Methods in Ecology and Evolution* 4.7 (2013), pp. 675–683.

[46] X. Dong and J. Jia. "Advances in Automatic Bird Species Recognition from Environmental Audio". In: *Journal of Physics: Conference Series* 1544.1 (2020). DOI: 10.1088/1742-6596/1544/1/012110.

[47] Robert Dooling. "Audition: Can Birds Hear Everything they Sing?" In: *Nature's Music: The Science of Birdsong*. Ed. by P. Marler and H Slabbekoor. 2004, pp. 206–225. ISBN: 978-0-080-47355-0.

[48] Guillaume Doras and Geoffroy Peeters. "A Prototypical Triplet Loss for Cover Detection". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 3797–3801. DOI: 10.1109/ICASSP40776.2020.9054619.

[49] Allison J Doupe and Patricia K Kuhl. "Birdsong and Human Speech: Common Themes and Mechanisms". In: *Annual review of neuroscience* 22.1 (1999), pp. 567–631.

[50] D.M. Endres and J.E. Schindelin. "A New Metric for Probability Distributions". In: *IEEE Transactions on Information Theory* 49.7 (2003), pp. 1858–1860. DOI: 10.1109/TIT.2003.813506.

[51] Mateus Espadoto, Rafael M Martins, Andreas Kerren, Nina ST Hirata, and Alexandru C Telea. "Toward a Quantitative Survey of Dimension Reduction

Techniques". In: *IEEE Transactions on Visualization and Computer Graphics* 27.3 (2019), pp. 2153–2173.

[52] S. Fagerlund. "Bird Species Recognition using Support Vector Machines". In: *Eurasip Journal on Advances in Signal Processing* 2007 (2007). DOI: 10.1155/2007/38637.

[53] S. Fagerlund and A. Härmä. "Parametrization of Inharmonic Bird sounds for Automatic Recognition". In: *13th European Signal Processing Conference, EUSIPCO 2005* (2005), pp. 1039–1042.

[54] Li Fei-Fei, Robert Fergus, and Pietro Perona. "One-shot Learning of Object Categories". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.4 (2006), pp. 594–611.

[55] Wolfgang Fiedler. "Chapter 9 - Bird Ecology as an Indicator of Climate and Global Change". In: *Climate Change*. Ed. by Trevor M. Letcher. Elsevier, 2009, pp. 181–195. ISBN: 978-0-444-53301-2. DOI: https://doi.org/10.1016/B978-0-444-53301-2.00009-9.

[56] Michael Fink. "Object Classification from a Single Example Utilizing Class Relevance Metrics". In: *Advances in Neural Information Processing Systems* 17 (2004).

[57] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-learning for Fast Adaptation of Deep Networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135.

[58] Quchen Fu, Zhongwei Teng, Jules White, Maria E. Powell, and Douglas C. Schmidt. "FastAudio: A Learnable Audio Front-End For Spoof Speech Detection". In: *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 3693–3697. DOI: 10.1109/ICASSP43922.2022.9746722.

[59] John Garofolo, L Lamel, W Fisher, Jonathan Fiscus, D Pallett, and Nancy Dahlgren. *DARPA-TIMIT Acoustic-Phonetic Continuous Speech Corpus*. en. Feb. 1993.

[60] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. "Audio Set: An Ontology and Human-labeled Dataset for Audio Events". In: *2017 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP). 2017, pp. 776–780. DOI: 10.1109/ICASSP.2017.7952261.

[61] H Glotin, Y LeCun, T Artieres, S Mallat, O Tchernichovski, and X Halkias. "Neural Information Processing Scaled for Bioacoustics, from Neurons to Big Data". In: *Workshop*. 2013.

[62] Hervé Goëau, Hervé Glotin, Willem-Pier Vellinga, Robert Planqué, and Alexis Joly. "LifeCLEF Bird Identification Task 2016: The Arrival of Deep Learning". In: *CLEF: Conference and Labs of the Evaluation Forum*. 1609. 2016, pp. 440–449.

[63] Franz Goller and Tobias Riede. "Integrative Physiology of Fundamental Frequency Control in Birds". In: *Journal of Physiology-Paris* 107.3 (2013), pp. 230–242.

[64] Robert Gove, Lucas Cadalzo, Nicholas Leiby, Jedediah M. Singer, and Alexander Zaitzeff. "New Guidance for using t-SNE: Alternative Defaults, Hyperparameter Selection Automation, and Comparative Evaluation". In: *Visual Informatics* 6.2 (2022), pp. 87–97. ISSN: 2468-502X. DOI: https://doi.org/10.1016/j.visinf.2022.04.003.

[65] Martin Graciarena, Michelle Delplanche, Elizabeth Shriberg, Andreas Stolcke, and Luciana Ferrer. "Acoustic Front-end Optimization for Bird Species Recognition". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 293–296. DOI: 10.1109/ICASSP.2010.5495923.

[66] Donald D Greenwood. "The Mel Scale's Disqualifying Bias and a Consistency of Pitch-difference Equisections in 1956 with Equal Cochlear Distances and Equal Frequency Ratios". In: *Hearing research* 103.1-2 (1997), pp. 199–224.

[67] T. Grill and J. Schluter. "Two Convolutional Neural Networks for Bird Detection in Audio Signals". In: *25th European Signal Processing Conference, EUSIPCO 2017* 2017-January (2017), pp. 1764–1768. DOI: 10.23919/EUSIPCO.2017.8081512.

[68] W. Halfwerk and H. Slabbekoorn. "A Behavioural Mechanism Explaining Noise-dependent Frequency Use in Urban Birdsong". In: *Animal Behaviour* 78.6 (2009), pp. 1301–1307.

[69] F.J. Harris. "On the use of Windows for Harmonic Analysis with the Discrete Fourier Transform". In: *Proceedings of the IEEE* 66.1 (1978), pp. 51–83. DOI: 10.1109/PROC.1978.10837.

[70] N. Harte, S. Murphy, D.J. Kelly, and N.M. Marples. "Identifying New Bird Species from Differences in Birdsong". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (2013), pp. 2900–2904.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[72] Hynek Hermansky. "Perceptual Linear Predictive (PLP) Analysis of Speech". In: *The Journal of the Acoustical Society of America* 87.4 (Apr. 1990), pp. 1738–1752. ISSN: 0001-4966. DOI: 10.1121/1.399423.

[73] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. "Speech Acoustic Modeling from Raw Multichannel Waveforms". In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 4624–4628.

[74] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[75] Yan Huang, Shang Li, Liang Wang, Tieniu Tan, et al. "Unfolding the Alternating Optimization for Blind Super Resolution". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5632–5643.

[76] G. Irvine, M. Cand, B. Davis, D. Coles, S. Miller, T. Levet, J. Shelton, J. Bass, D. Sexton, and G Leventhall. *IOA Noise Working Group (Wind Turbine Noise), Amplitude Modulation Working Group: A Method for Rating Amplitude Modulation in Wind Turbine Noise*. 2016.

[77] Navdeep Jaitly and Geoffrey Hinton. "Learning a Better Representation of Speech Soundwaves using Restricted Boltzmann Machines". In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 5884–5887.

[78] P. Jancovic and M. Kokuer. "Automatic Detection of Bird Species from Audio Field Recordings using HMM-based Modelling of Frequency Tracks". In: *25th European Signal Processing Conference, EUSIPCO 2017* 2017-January (2017), pp. 1779–1783. DOI: 10.23919/EUSIPCO.2017.8081515.

[79] Peter Jancovic and Munevver Kokuer. "Automatic Detection and Recognition of Tonal Bird Sounds in Noisy Environments". In: *EURASIP J. Adv. Sig. Proc.* 2011 (Jan. 2011). DOI: 10.1155/2011/982936.

[80] Peter Jancovic and Munevver Kokuer. "Bird Species Recognition using Unsupervised Modeling of Individual Vocalization Elements". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27 (2019), pp. 932–947.

[81] Peter Jancovic and Munevver Kokuer. "Detection of Sinusoidal Signals in Noise by Probabilistic Modelling of the Spectral Magnitude, Shape and Phase Continuity". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. June 2011, pp. 517–520. DOI: 10.1109/ICASSP.2011.5946454.

[82] Jens Johannsmeier and Sebastian Stober. *Few-Shot Bioacoustic Event Detection via Segmentation using Prototypical Networks*. Tech. rep. DCASE2021 Challenge, June 2021.

[83] A. Johnston, M. Ausden, A.M. Dodd, R.B. Bradbury, D.E. Chamberlain, F. Jiguet, C.D. Thomas, A.S.C.P. Cook, S.E. Newson, N. Ockendon, M.M. Rehfisch, S. Roos, C.B. Thaxter, A. Brown, H.Q.P. Crick, A. Douse, R.A. McCall, H. Pontier, D.A. Stroud, B. Cadiou, O. Crowe, B. Deceuninck, M. Hornman, and J.W. Pearce-Higgins. "Observed and Predicted Effects of Climate Change on Species Abundance in Protected Areas". In: *Nature Climate Change* 3.12 (2013), pp. 1055–1061. DOI: 10.1038/nclimate2035.

[84] Stefan Kahl, Tom Denton, Holger Klinck, Hervé Glotin, Hervé Goëau, Willem-Pier Vellinga, Robert Planqué, and Alexis Joly. "Overview of BirdCLEF 2021: Bird Call Identification in Soundscape Recordings". In: *Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum*. 2021.

[85] Stefan Kahl, Tom Denton, Holger Klinck, Hendrik Reers, Francis Cherutich, Hervé Glotin, Hervé Goëau, Willem-Pier Vellinga, Robert Planqué, and Alexis Joly. "Overview of BirdCLEF 2023: Automated bird species identification in Eastern Africa". In: *Working Notes of CLEF* (2023).

[86] Stefan Kahl, Amanda Navine, Tom Denton, Holger Klinck, Patrick Hart, Hervé Glotin, Hervé Goëau, Willem-Pier Vellinga, Robert Planqué, and Alexis Joly. "Overview of BirdCLEF 2022: Endangered Bird Species Recognition in Soundscape Recordings". In: *Working Notes of CLEF* (2022).

[87] Stefan Kahl, Connor M. Wood, Maximilian Eibl, and Holger Klinck. "BirdNET: A Deep Learning Solution for Avian Diversity Monitoring". In: *Ecological Informatics* 61 (2021), p. 101236. ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2021.101236.

[88] Terri Kamm, Hynek Hermansky, and Andreas Andreou. "Learning the Mel-scale and Optimal VTN Mapping". In: *Technical Report, JHU/CLSP Workshop* (1997).

[89] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. "Supervised Contrastive Learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18661–18673.

[90] Diederik P Kingma and Jimmy Ba. "ADAM: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[91] Tomi Kinnunen and Haizhou Li. "An Overview of Text-independent Speaker Recognition: From Features to Supervectors". In: *Speech Communication* 52.1 (2010), pp. 12–40. DOI: https://doi.org/10.1016/j.specom.2009.08.009.

[92] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. "Siamese Neural Networks for One-shot Image Recognition". In: *ICML deep learning workshop*. Vol. 2. 1. Lille. 2015.

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25.

Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[94] Steinar Laenen and Luca Bertinetto. "On Episodes, Prototypical Networks, and Few-Shot Learning". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 24581–24592. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/cdfa4c42f465a5a66871587c69fcfa34-Paper.pdf.

[95] R. Laje and G. B. Mindlin. "Modeling Source-Source and Source-Filter Acoustic Interaction in Birdsong". In: *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 72.3 (2005).

[96] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. "One Shot Learning of Simple Visual Concepts". In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 33. 33. 2011.

[97] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. "Building Machines that Learn and Think like People". In: *Behavioral and brain sciences* 40 (2017), e253.

[98] M. Lasseck. "Acoustic Bird Detection with Deep Convolutional Neural Networks". In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop* (2018). URL: http://dcase.community/documents/challenge2018/technical_reports/DCASE2018_Lasseck_76.pdf.

[99] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. "Meta-learning with Differentiable Convex Optimization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10657–10665.

[100] Younglo Lee, Jeongki Min, David K. Han, and Hanseok Ko. "Spectro-Temporal Attention-Based Voice Activity Detection". In: *IEEE Signal Processing Letters* 27 (2020), pp. 131–135. DOI: 10.1109/LSP.2019.2959917.

[101] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365.

[102] Feng Li, Yixuan Wu, Huihui Bai, Weisi Lin, Runmin Cong, Chunjie Zhang, and Yao Zhao. "Learning Detail-Structure Alternative Optimization for Blind Super-Resolution". In: *IEEE Transactions on Multimedia* (2022).

[103] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. "Meta-SGD: Learning to Learn Quickly for Few-shot Learning". In: *arXiv preprint arXiv:1707.09835* (2017).

[104] Sidrah Liaqat, Narjes Bozorg, Neenu Jose, Patrick Conrey, Antony Tamasi, and Michael T. Johnson. *Domain Tuning Methods for Bird Audio Detection*. Tech. rep. DCASE2018 Challenge, Sept. 2018.

[105] Haohe Liu, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Wenwu Wang, and Mark D Plumbley. *Surrey System for DCASE 2022 Task 5: Few-Shot Bioacoustic Event Detection with Segment-Level Metric Learning*. Tech. rep. DCASE2022 Challenge, June 2022.

[106] Junyan Liu, Zikai Zhou, Mengkai Sun, Kele Xu, Kun Qian, and Bian Hu. *SE-Protonet: Prototypical Network with Squeeze-and-Excitation Blocks for Bioacoustic Event Detection*. Tech. rep. DCASE2023 Challenge, June 2023.

[107] Xuechen Liu, Md Sahidullah, and Tomi Kinnunen. "Learnable MFCCs for Speaker Verification". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2021, pp. 1–5.

[108] Xuechen Liu, Md Sahidullah, and Tomi Kinnunen. "Learnable Nonlinear Compression for Robust Speaker Verification". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 7962–7966.

[109] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sungju Hwang, and Yi Yang. "Learning to Propagate Labels: Transductive Propagation Network for Few-shot Learning". In: *International Conference on Learning Representations*. 2019.

[110] Iván López-Espejo, Ram C. M. C. Shekar, Zheng-Hua Tan, Jesper Jensen, and John H. L. Hansen. "Filterbank Learning for Noise-Robust Small-Footprint Keyword Spotting". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095436.

[111] Iván López-Espejo, Zheng-Hua Tan, and Jesper Jensen. "Exploring Filterbank Learning for Keyword Spotting". In: *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 331–335.

[112] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. arXiv:1608.03983. 2016. DOI: 10.48550/ARXIV.1608.03983. URL: https://arxiv.org/abs/1608.03983.

[113] Vincent Lostanlen, Antoine Bernabeu, Jean-Luc Béchennec, Mikaël Briday, Sébastien Faucou, and Mathieu Lagrange. "Energy Efficiency is Not Enough: Towards a Batteryless Internet of Sounds". In: *Proceedings of the 16th International Audio Mostly Conference*. AM '21. virtual/Trento, Italy: Association for Computing Machinery, 2021, pp. 147–155. ISBN: 9781450385695. DOI: 10.1145/3478384.3478408.

[114] Vincent Lostanlen, Kaitlin Palmer, Elly Knight, Christopher Clark, Holger Klinck, Andrew Farnsworth, Tina Wong, Jason Cramer, and Juan Bello. "Long-distance Detection of Bioacoustic Events with Per-channel Energy Normalization". In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York University, NY, USA, Oct. 2019, pp. 144–148.

[115] Vincent Lostanlen, Justin Salamon, Mark Cartwright, Brian McFee, Andrew Farnsworth, Steve Kelling, and Juan Pablo Bello. "Per-Channel Energy Normalization: Why and How". In: *IEEE Signal Processing Letters* 26.1 (2019), pp. 39–43. DOI: 10.1109/LSP.2018.2878620.

[116] Vincent Lostanlen, Justin Salamon, Andrew Farnsworth, Steve Kelling, and Juan Pablo Bello. "BirdVox-Full-Night: a Dataset and Benchmark for Avian Flight Call Detection". In: *Proc. IEEE ICASSP* (Calgary, Canada). Apr. 2018.

[117] Vincent Lostanlen, Justin Salamon, Andrew Farnsworth, Steve Kelling, and Juan Pablo Bello. "Robust Sound Event Eetection in Bioacoustic Sensor Networks". In: *PLOS ONE* 14.10 (Oct. 2019), pp. 1–31. DOI: 10.1371/journal.pone.0214168.

[118] Xu Luo, Hao Wu, Ji Zhang, Lianli Gao, Jing Xu, and Jingkuan Song. "A Closer Look at Few-shot Classification Again". In: *arXiv preprint arXiv:2301.12246* (2023).

[119] James G Lyons and Kuldip K Paliwal. "Effect of Compressing the Dynamic Range of the Power Spectrum in Modulation Filtering-based Speech Enhancement". In: *Ninth Annual Conference of the International Speech Communication Association*. 2008.

[120] Peter R. Marler. "A Comparative Approach to Vocal Learning: Song Development in White-Crowned Sparrows". In: *Journal of Comparative and Physiological Psychology* 71 (1970), pp. 1–25.

[121] Tiago A. Marques, Len Thomas, Stephen W. Martin, David K. Mellinger, Jessica A. Ward, David J. Moretti, Danielle Harris, and Peter L. Tyack. "Estimating Animal Population Density using Passive Acoustics". In: *Biological Reviews* 88.2 (2013), pp. 287–309. DOI: https://doi.org/10.1111/brv.12001.

[122] John Martinsson, Martin Willbo, Aleksis Pirinen, Olof Mogren, and Maria Sandsten. *Few-Shot Bioacoustic Event Detection using a Prototypical Network Ensemble with Adaptive Embedding Functions*. Tech. rep. DCASE2022 Challenge, June 2022.

[123] Volodymyr Melnykov and Igor Melnykov. "Initializing the EM algorithm in Gaussian Mixture Models with an Unknown Number of Components". In: *Computational Statistics & Data Analysis* 56.6 (2012), pp. 1381–1395. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2011.11.002.

[124] V. Morfi, Y. Bas, H. Pamula, H. Glotin, and D. Stowell. "NIPS4Bplus: A Richly Annotated Birdsong Audio Dataset". In: *PeerJ Computer Science* 2019.10 (2019). DOI: 10.7717/peerj-cs.223.

[125] Veronica Morfi, Ines Nolasco, Vincent Lostanlen, Shubhr Singh, Ariana Strandburg-Peshkin, Lisa Gill, Hanna Pamuła, David Benvent, and Dan Stowell. "Few-Shot Bioacoustic Event Detection: A New Task at the DCASE

2021 Challenge". In: *Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*. Barcelona, Spain, Nov. 2021, pp. 145–149. ISBN: 978-84-09-36072-7.

[126] Ilyass Moummad, Romain Serizel, and Nicolas Farrugia. *Supervised Contrastive Learning for Pre-Training Bioacoustic Few Shot Systems*. Tech. rep. DCASE2023 Challenge, June 2023.

[127] Leah Mutanu, Jeet Gohil, Khushi Gupta, Perpetua Wagio, and Gerald Kotonya. "A Review of Automated Bioacoustics and General Acoustics Classification Research". In: *Sensors* 22.21 (2022), p. 8361.

[128] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. "Speech Recognition using Deep Neural Networks: A Systematic Review". In: *IEEE Access* 7 (2019), pp. 19143–19165. DOI: 10.1109/ACCESS.2019.2896880.

[129] Paul-Gauthier Noé, Titouan Parcollet, and Mohamed Morchid. "CGCNN: Complex Gabor Convolutional Neural Network on Raw Speech". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7724–7728.

[130] Ines Nolasco, S Singh, E Vidana-Villa, E Grout, J Morford, M Emmerson, F Jensens, H Whitehead, Ivan Kiskin, A Strandburg-Peshkin, et al. "Few-shot Bioacoustic Event Detection at the DCASE 2022 Challenge". In: *arXiv preprint arXiv:2207.07911* (2022).

[131] Inês Nolasco, Shubhr Singh, Veronica Morfi, Vincent Lostanlen, Ariana Strandburg-Peshkin, Ester Vidaña-Vila, Lisa Gill, Hanna Pamuła, Helen Whitehead, Ivan Kiskin, et al. "Learning to Detect an Animal Sound from Five Examples". In: *Ecological Informatics* 77 (2023), p. 102258.

[132] S. Nowicki. "Vocal Tract Resonances in Oscine Bird Sound Production: Evidence from Birdsongs in a Helium Atmosphere". In: *Nature* 325.6099 (1987), pp. 53–55.

[133] C. O'Reilly, K. Analuddin, D. J. Kelly, and N. Harte. "Measuring Vocal Difference in Bird Population Pairs". In: *Journal of the Acoustical Society of America* 143.3 (2018), pp. 1658–1671.

[134] C. O'Reilly, N.M. Marples, D.J. Kelly, and N. Harte. "YIN-bird: Improved Pitch Tracking for Bird Vocalisations". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 08-12-September-2016 (2016), pp. 2641–2645. DOI: 10.21437/Interspeech.2016-90.

[135] D. O'Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Publishing Company, 1987. ISBN: 9780201165203.

[136] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. "TADAM: Task Dependent Adaptive Metric for Improved Few-shot Learning". In: *Advances in Neural Information Processing Systems* 31 (2018).

[137] Dimitri Palaz, Ronan Collobert, and Mathew Magimai-Doss. "Estimating Phoneme Class Conditional Probabilities from Raw Speech Signal using Convolutional Neural Networks". In: *Proc. Interspeech 2013*. 2013, pp. 1766–1770. DOI: 10.21437/Interspeech.2013-438.

[138] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Interspeech 2019* (Sept. 2019). DOI: 10.21437/interspeech.2019-2680.

[139] Archit Parnami and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-shot Learning". In: *arXiv preprint arXiv:2203.04291* (2022).

[140] Alberto García Arroba Parrilla and Dan Stowell. "Polyphonic Sound Event Detection for Highly Dense Birdsong Scenes". In: *arXiv preprint arXiv:2207.06349* (2022).

[141] Karl Pearson. *LIII. On Lines and Planes of Closest Fit to Systems of Points in Space*. Nov. 1901. DOI: 10.1080/14786440109462720.

[142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine

Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[143] Junyi Peng, Rongzhi Gu, Ladislav Mošner, Oldrich Plchot, Lukás Burget, and Jan Černocký. "Learnable Sparse Filterbank for Speaker Verification". In: *Proc. Interspeech 2022*. 2022, pp. 5110–5114.

[144] Karol J. Piczak. "ESC: Dataset for Environmental Sound Classification". In: *Proceedings of the 23rd Annual ACM Conference on Multimedia*. Brisbane, Australia: ACM Press, Oct. 13, 2015, pp. 1015–1018. ISBN: 978-1-4503-3459-4. DOI: 10.1145/2733373.2806390.

[145] J. Podos. "A Performance Constraint on the Evolution of Trilled Vocalizations in a Songbird Family (Passeriformes: Emberizidae)". In: *Evolution* 51.2 (1997), pp. 537–551.

[146] Jeffery Podos and Stephen Nowicki. "Performance Limits on Birdsong". In: *Nature's Music: The Science of Birdsong*. Ed. by P. Marler and H Slabbekoor. 2004, pp. 318–342. ISBN: 978-0-080-47355-0.

[147] Jordi Pons, Joan Serrà, and Xavier Serra. "Training Neural Audio Classifiers with Few Data". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 16–20.

[148] Nirosha Priyadarshani, Stephen Marsland, and Isabel Castro. "Automated Birdsong Recognition in Complex Acoustic Environments: A Review". In: *Journal of Avian Biology* 49.5 (2018), jav–01447. DOI: https://doi.org/10.1111/jav.01447.

[149] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[150] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. "MIR_EVAL: A Transparent Implementation of Common MIR Metrics." In: *ISMIR*. Vol. 10. 2014, p. 2014.

[151] Mirco Ravanelli and Yoshua Bengio. "Speaker Recognition from Raw Waveform with SincNet". In: *2018 IEEE Spoken Language Technology Workshop (SLT)*. 2018, pp. 1021–1028. DOI: 10.1109/SLT.2018.8639585.

[152] Sachin Ravi and Hugo Larochelle. "Optimization as a Model for Few-shot Learning". In: *International conference on learning representations*. 2016.

[153] Chandan KA Reddy, Ebrahim Beyrami, Jamie Pool, Ross Cutler, Sriram Srinivasan, and Johannes Gehrke. "A Scalable Noisy Speech Dataset and Online Subjective Test Framework". In: *Proc. Interspeech 2019* (2019), pp. 1816–1820.

[154] R. S. Rempel, C. M. Francis, J. N. Robinson, and M. Campbell. "Comparison of Audio Recording System Performance for Detecting and Monitoring Songbirds". In: *Journal of Field Ornithology* 84.1 (2013), pp. 86–97. DOI: 10.1111/jofo.12008.

[155] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. "Meta-learning for Semi-supervised Few-shot Classification". In: *arXiv preprint arXiv:1803.00676* (2018).

[156] Frank E Rheindt, Janette A Norman, and Les Christidis. "DNA Evidence shows Vocalizations to be a better indicator of Taxonomic Limits than Plumage Patterns in Zimmerius Tyrant-Flycatchers". In: *Molecular Phylogenetics and Evolution* 48.1 (2008), pp. 150–156.

[157] Rachid Riad, Julien Karadayi, Anne-Catherine Bachoud-Lévi, and Emmanuel Dupoux. "Learning Spectro-Temporal Representations of Complex Sounds with Parameterized Neural Networks". In: *The Journal of the Acoustical Society of America* 150.1 (2021), pp. 353–366. DOI: 10.1121/10.0005482.

[158] T. Riede, R. A. Suthers, N. H. Fletcher, and W. E. Blevins. "Songbirds Tune their Vocal Tract to the Fundamental Frequency of their Song". In: *Proceedings of the National Academy of Sciences of the United States of America* 103.14 (2006), pp. 5543–5548.

[159] Pau Rodríguez, Issam Laradji, Alexandre Drouin, and Alexandre Lacoste. *Embedding Propagation: Smoother Manifold for Few-Shot Classification*. 2020. arXiv: 2003.04151 [cs.CV].

[160] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, and Bhuvana Ramabhadran. "Learning Filter Banks within a Deep Neural Network Framework". In: *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE. 2013, pp. 297–302.

[161] Tara N. Sainath, Brian Kingsbury, Abdel-rahman Mohamed, and Bhuvana Ramabhadran. "Learning Filter Banks within a Deep Neural Network Framework". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. 2013, pp. 297–302. DOI: 10.1109/ASRU.2013.6707746.

[162] Tara N. Sainath, Ron J. Weiss, Andrew Senior, Kevin W. Wilson, and Oriol Vinyals. "Learning the Speech Front-end with Raw Waveform CLDNNs". In: *Proc. Interspeech 2015*. 2015, pp. 1–5. DOI: 10.21437/Interspeech.2015-1.

[163] Justin Salamon, Juan Pablo Bello, Andrew Farnsworth, Matt Robbins, Sara Keen, Holger Klinck, and Steve Kelling. "Towards the Automatic Classification of Avian Flight Calls for Bioacoustic Monitoring". In: *PLOS ONE* 11.11 (Nov. 2016), pp. 1–26. DOI: 10.1371/journal.pone.0166866.

[164] Susanta Sarangi, Md Sahidullah, and Goutam Saha. "Optimization of Data-driven Filterbank for Automatic Speaker Verification". In: *Digital Signal Processing* 104 (2020), p. 102795. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2020.102795.

[165] R. Schluter, I. Bezrukov, H. Wagner, and H. Ney. "Gammatone Features and Feature Combination for Large Vocabulary Speech Recognition". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 4. 2007, pp. IV-649-IV–652. DOI: 10.1109/ICASSP.2007.366996.

[166] Jan Schlüter and Gerald Gutenbrunner. *EfficientLEAF: A Faster LEarnable Audio Frontend of Questionable Use*. arXiv.2207.05508. 2022. DOI: 10.48550/ARXIV.2207.05508. URL: https://arxiv.org/abs/2207.05508.

[167] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *2015 IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR). 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[168] Hiroshi Seki, Kazumasa Yamamoto, and Seiichi Nakagawa. "A Deep Neural Network Integrated with Filterbank Learning for Speech Recognition". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 5480–5484.

[169] Changhao Shan, Junbo Zhang, Yujun Wang, and Lei Xie. *Attention-based End-to-End Models for Small-Footprint Keyword Spotting*. 2018. arXiv: 1803.10916 [cs.SD].

[170] S. S. Shapiro and M. B. Wilk. "An Analysis of Variance Test for Normality". In: *Biometrics* 52.3-4 (Dec. 1965), pp. 591–611. ISSN: 0006-3444. DOI: 10.1093/biomet/52.3-4.591.

[171] G. Sharma, K. Umapathy, and S. Krishnan. "Trends in Audio Signal Feature Extraction Methods". In: *Applied Acoustics* 158 (2020). DOI: 10.1016/j.apacoust.2019.107020.

[172] Bowen Shi, Ming Sun, Krishna C Puvvada, Chieh-Chi Kao, Spyros Matsoukas, and Chao Wang. "Few-shot Acoustic Event Detection via Meta Learning". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 76–80.

[173] Roman Shrestha, Cornelius Glackin, Julie Wall, and Nigel Cannings. "Bird Audio Diarization with Faster R-CNN". In: *International Conference on Artificial Neural Networks*. Springer. 2021, pp. 415–426.

[174] H. Slabbekoorn. "Songs of the City: Noise-dependent Spectral Plasticity in the Acoustic Phenotype of Urban Birds". In: *Animal Behaviour* 85.5 (2013), pp. 1089–1099.

[175] H. Slabbekoorn and M. Peet. "Birds Sing at a Higher Pitch in Urban Noise". In: *Nature* 424.6946 (2003), p. 267.

[176] Evan C Smith and Michael S Lewicki. "Efficient Auditory Coding". In: *Nature* 439.7079 (2006), pp. 978–982.

[177] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical Networks for Few-shot Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

[178] Jongseo Sohn, Nam Soo Kim, and Wonyong Sung. "A Statistical Model-based Voice Activity Detection". In: *IEEE Signal Processing Letters* 6.1 (1999), pp. 1–3. DOI: 10.1109/97.736233.

[179] J. Stastny, M. Munk, and L. Juranek. "Automatic Bird Species Recognition based on Birds Vocalization". In: *Eurasip Journal on Audio, Speech, and Music Processing* 2018.1 (2018). DOI: 10.1186/s13636-018-0143-7.

[180] Stanley S Stevens and John Volkmann. "The Relation of Pitch to Frequency: A Revised Scale". In: *The American Journal of Psychology* 53.3 (1940), pp. 329–353.

[181] Thomas G. Stockham. "High-Speed Convolution and Correlation". In: *Proceedings of the April 26-28, 1966, Spring Joint Computer Conference*. AFIPS '66 (Spring). Boston, Massachusetts: Association for Computing Machinery, 1966, pp. 229–233. ISBN: 9781450378925. DOI: 10.1145/1464182.1464209.

[182] D. Stowell, L. Gill, and D. Clayton. "Detailed Temporal Structure of Communication Networks in Groups of Songbirds". In: *Journal of the Royal Society Interface* 13.119 (2016).

[183] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin. "Bird Detection in Audio: A Survey and a Challenge". In: *IEEE International Workshop on Machine Learning for Signal Processing, MLSP* 2016-November (2016). DOI: 10.1109/MLSP.2016.7738875.

[184] Dan Stowell. *Computational Bioacoustics with Deep Learning: a Review and Roadmap*. 2021. arXiv: 2112.06725 [cs.SD].

[185] Dan Stowell and Mark D. Plumbley. "An Open Dataset for Research on Audio Field Recording Archives: freefield1010". In: *CoRR* abs/1309.5275 (2013). arXiv: 1309.5275. URL: http://arxiv.org/abs/1309.5275.

[186] Dan Stowell and Mark D. Plumbley. "Automatic Large-scale Classification of Bird Sounds is Strongly Improved by Unsupervised Feature Learning". In: *PeerJ* 2 (July 2014), e488. ISSN: 2167-8359. DOI: 10.7717/peerj.488.

[187] Dan Stowell, Michael D. Wood, Hanna Pamuła, Yannis Stylianou, and Hervé Glotin. "Automatic Acoustic Detection of Birds through Deep Learning: The First Bird Audio Detection Challenge". In: *Methods in Ecology and Evolution* 10.3 (2019), pp. 368–380. DOI: https://doi.org/10.1111/2041-210X.13103.

[188] Larissa Sayuri Moreira Sugai, Thiago Sanna Freire Silva, Jr Ribeiro José Wagner, and Diego Llusia. "Terrestrial Passive Acoustic Monitoring: Review and Perspectives". In: *BioScience* 69.1 (Nov. 2018), pp. 15–25. ISSN: 0006-3568. DOI: 10.1093/biosci/biy147.

[189] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. "Learning to Compare: Relation Network for Few-shot Learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1199–1208.

[190] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[191] Jigang Tang, Zhang Xueyang, Tian Gao, Diyuan Liu, Xin Fang, Jia Pan, Qing Wang, Jan Du, Kele Xu, and Qinghua Pan. *Few-Shot Embedding Learning and Event Filtering for Bioacoustic Event Detection*. Tech. rep. DCASE2022 Challenge, June 2022.

[192] Tiantian Tang, Yunhao Liang, and Yanhua Long. *Two Improved Architectures Based on Prototype Network for Few-Shot Bioacoustic Event Detection*. Tech. rep. DCASE2021 Challenge, June 2021.

[193] Nicholas S. Thompson, Kerry Ledoux, and Kevin Moody. "A System for Describing Bird Song Units". In: *Bioacoustics-the International Journal of Animal Sound and Its Recording* 5 (1994), pp. 267–279.

[194] W. H. Thorpe. "The Process of Song-learning in the Chaffinch as studied by Means of the Sound Spectrograph". In: *Nature* 173.4402 (1954), pp. 465–469.

[195] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. "Rethinking Few-shot Image Classification: A Good Embedding is All You Need?" In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*. Springer. 2020, pp. 266–282.

[196] D. Tilman, M. Clark, D.R. Williams, K. Kimmel, S. Polasky, and C. Packer. "Future Threats to Biodiversity and Pathways to their Prevention". In: *Nature* 546.7656 (2017), pp. 73–81. DOI: 10.1038/nature22900.

[197] Joseph A Tobias, Nathalie Seddon, Claire N Spottiswoode, John D Pilgrim, Lincoln DC Fishpool, and Nigel J Collar. "Quantitative Criteria for Species Delimitation". In: *Ibis* 152.4 (2010), pp. 724–746.

[198] Hartmut Traunmüller. "Analytical Expressions for the Tonotopic Sensory Scale". In: *The Journal of the Acoustical Society of America* 88.1 (1990), pp. 97–100. DOI: 10.1121/1.399849.

[199] John W. Tukey. "Comparing Individual Means in the Analysis of Variance". In: *Biometrics* 5.2 (1949), pp. 99–114. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/3001913 (visited on 05/30/2023).

[200] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[201] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All You Need". In: *Advances in Neural Information Processing Systems* 30 (2017).

[202] Olivier Veilleux, Malik Boudiaf, Pablo Piantanida, and Ismail Ben Ayed. "Realistic Evaluation of Transductive Few-shot Learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 9290–9302.

[203] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu koray, and Daan Wierstra. "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

[204] Peter M Vitousek, John D Aber, Robert W Howarth, Gene E Likens, Pamela A Matson, David W Schindler, William H Schlesinger, and David G Tilman. "Human Alteration of the Global Nitrogen Cycle: Sources and Consequences". In: *Ecological applications* 7.3 (1997), pp. 737–750.

[205] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. "Low-shot Learning from Imaginary Data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7278–7286.

[206] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens Van Der Maaten. "Simpleshot: Revisiting Nearest-neighbor Classification for Few-shot Learning". In: *arXiv preprint arXiv:1911.04623* (2019).

[207] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM computing surveys (csur)* 53.3 (2020), pp. 1–34.

[208] Yu Wang, Nicholas J. Bryan, Justin Salamon, Mark Cartwright, and Juan Pablo Bello. "Who Calls The Shots? Rethinking Few-Shot Learning for Audio". In: *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 2021, pp. 36–40. DOI: 10.1109/WASPAA52581.2021.9632677.

[209] Yu Wang, Justin Salamon, Nicholas J Bryan, and Juan Pablo Bello. "Few-shot Sound Event Detection". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 81–85.

[210] Yu Wang, Justin Salamon, Nicholas J. Bryan, and Juan Pablo Bello. "Few-Shot Sound Event Detection". In: *ICASSP 2020 - 2020 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP). 2020, pp. 81–85. DOI: 10.1109/ICASSP40776.2020.9054708.

[211] Yuxuan Wang, Pascal Getreuer, Thad Hughes, Richard F. Lyon, and Rif A. Saurous. "Trainable Frontend for Robust and Far-field Keyword Spotting". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 5670–5674. DOI: 10.1109/ICASSP.2017.7953242.

[212] David H. Wolpert. "Stacked Generalization". In: *Neural Networks* 5.2 (1992), pp. 241–259. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80023-1.

[213] Genwei Yan, Ruoyu Wang, Liang Zou, Jun Du, Qing Wang, Tian Gao, and Xin Fang. *Multi-Task Frame Level System for Few-Shot Bioacoustic Event Detection*. Tech. rep. DCASE2023 Challenge, June 2023.

[214] Dongchao Yang, Helin Wang, Zhongjie Ye, and Yuexian Zou. *Few-Shot Bioacoustic Event Detection = A Good Transductive Inference is All You Need*. Tech. rep. DCASE2021 Challenge, June 2021.

[215] Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi. "TorchAudio: Building Blocks for Audio and Speech Processing". In: *arXiv preprint arXiv:2110.15018* (2021).

[216] Neil Zeghidour, Olivier Teboul, Felix de Chaumont Quitry, and Marco Tagliasacchi. "LEAF: A Learnable Frontend for Audio Classification". In: *CoRR* abs/2101.08596 (2021). arXiv: 2101.08596. URL: https://arxiv.org/abs/2101.08596.

[217] Neil Zeghidour, Nicolas Usunier, Iasonas Kokkinos, Thomas Schaiz, Gabriel Synnaeve, and Emmanuel Dupoux. "Learning Filterbanks from Raw Speech for Phone Recognition". In: *2018 IEEE International Conference on Acoustics, Speech and*

Signal Processing (ICASSP). 2018, pp. 5509–5513. DOI: 10.1109/ICASSP.2018.8462015.

[218] Neil Zeghidour, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert, and Emmanuel Dupoux. "End-to-End Speech Recognition from the Raw Waveform". In: *arXiv preprint arXiv:1806.07098* (2018).

[219] Jinshan Zeng, Tim Tsz-Kit Lau, Shaobo Lin, and Yuan Yao. "Global Convergence of Block Coordinate Descent in Deep Learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7313–7323.

[220] Yue Zhang, Jun Wang, Dawei Zhang, and Feng Deng. *Few-Shot Bioacoustic Event Detection using Prototypical Network with Background Class*. Tech. rep. DCASE2021 Challenge, June 2021.