Universitat Pompeu Fabra
Barcelona

Interpretable Deep-learning Models for Sound Event Detection and Classification

Pablo Zinemanas Friet

TESI DOCTORAL UPF / 2023

Thesis supervisors:

Dr. Xavier Serra Casals
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona, Spain

Dr. Frederic Font Corbera
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona, Spain

Dissertation submitted to the Department of Information and Communication Technologies of Universitat Pompeu Fabra in partial fulfillment of the requirements for the degree of

DOCTOR PER LA UNIVERSITAT POMPEU FABRA

Music Technology Group (http://mtg.upf.edu), Dept. of Information and Communication Technologies (http://www.upf.edu/dtic), Universitat Pompeu Fabra (http://www.upf.edu), Barcelona, Spain.

This thesis has been carried out at the Music Technology Group (MTG) of Universitat Pompeu Fabra in Barcelona, Spain, from Sep. 2019 to Sep. 2022. It was supervised by Dr. Xavier Serra Casals and Dr. Frederic Font Corbera. The main content of this document was finished in Sep. 2022, so more recent articles are not referenced in the dissertation.

The work from Chapter 3 has been conducted in collaboration with Ignacio Hounie, Pablo Cancela, and Martín Rocamora from Universidad de la República, Uruguay. Work in Chapter 4 was also done in collaboration with Pablo Cancela and Martín Rocamora. The work in Chapters 5 and 6 has been done in close collaboration with Martín Rocamora, Eduardo Fonseca, and Marius Miron from Universitat Pompeu Fabra.

To Alba

Acknowledgements

I would like to start by expressing my gratitude to my supervisors, Dr. Xavier Serra Casals and Dr. Frederic Font; this work has not been possible without their support. I would also like to thank all the researchers I have collaborated with during these years, including Ignacio Hounie, Pablo Cancela, Eduardo Fonseca, Marius Miron, Genis Plaja, Juan Pablo Bello, Magdalena Fuentes, and Justin Salamon. I especially thank Martín Rocamora for his guidance, support, and mentoring since I started working on environmental sounds in 2016.

I thank Cristina Garrido, Sonia Espí, and Lydia García for all the help during these years in administrative tasks.

I also want to acknowledge all the members of the MTG that were with me at different moments during the last four years for their feedback, the discussions, and also for making this journey more pleasant. Thanks to Alastair Porter, Albin Correya, Alia Morsi, Antonio Ramirez, Benno Weck, Dmitry Bogdanov, Ernic Guiso, Jorge Marcos, Furkan Yesiler, Guillem Cortès, Hyon Kim, Lorenzo Porcaro, Luis Joglar, Jordi Pons, Juan Gómez, Marius Miron, Michael Taenzer, Miguel Perez, Pablo Alonso, Panagiota Anastasopoulou, Pedro Ramoneda, Philip Tovstogan, Roser Batlle, Vsevolod Eremenko, Xavier Favory and Xavier Lizarraga.

I thank my whole family, especially my parents Enrique and Eva, for always supporting me, even from a long distance. I would also like to thank my friends from Uruguay, particularly Claudia, Maite, and Martín, whose friendship and support have been there for many years. Finally, I thank Ana for her love, patience, support, and motivation. But above all, for sharing with me this journey in Barcelona during the great, but also hard, moments that we have experienced.

Abstract

Deep-learning models have revolutionized state-of-the-art technologies in many research areas, but their black-box structure makes it difficult to understand their inner workings and the rationale behind their predictions. This may lead to unintended effects, such as being susceptible to adversarial attacks or the reinforcement of biases. As a consequence, there has been an increasing interest in developing deep-learning models that provide explanations of their decisions, a field known as interpretable deep learning. On the other hand, in the past few years, there has been a surge in developing technologies for environmental sound recognition motivated by its applications in healthcare, smart homes, or urban planning. However, most of the systems used for these applications are deep-learning-based black boxes and, therefore, can not be inspected, so the rationale behind their decisions is obscure. Despite recent advances, there is still a lack of research in interpretable machine learning in the audio domain. This thesis aims to reduce this gap by proposing several interpretable deep-learning models for automatic sound classification and event detection.

We start by describing an open-source software tool for reproducible research in the sound recognition field, which was used to implement the models and run experiments presented in this document. We then propose an interpretable front-end based on domain knowledge to tailor the feature-extraction layers of an end-to-end network for sound event detection. We then present a novel interpretable deep-learning model for automatic sound classification, which explains its predictions based on the similarity of the input to a set of learned prototypes in a latent space. We leverage domain knowledge by designing a frequency-dependent similarity measure. The proposed model achieves results comparable to state-of-the-art methods. In addition, we present two automatic methods to prune the proposed model that exploits its interpretability. This model is accompanied by a web application for the manual editing of the model, which allows for a human-in-the-loop debugging approach. Finally, we propose an extension of this model that works for a polyphonic setting, such as the sound event detection task. To provide interpretability, we leverage the prototype network approach and attention mechanisms.

The tools for reproducible research and the interpretable deep-learning models, such as those proposed in this thesis, can contribute to developing a more responsible and trustworthy Artificial Intelligence in the audio domain.

Resum

Els models de *deep learning* han revolucionat les tecnologies d'última generació en moltes àrees de recerca, però la seva estructura *black-box* fa difícil entendre el seu funcionament intern i la lògica darrere de les seves prediccions. Això pot conduir a efectes no desitjats, com ara ser susceptible a atacs adversos o el reforç de biaixos. Com a conseqüència, hi ha hagut un interès creixent en el desenvolupament de models de *deep learning* que proporcionen explicacions de les seves decisions, un camp conegut com a *deep learning* interpretable. D'altra banda, en els últims anys, s'ha produït un augment en el desenvolupament de les tecnologies per al reconeixement de so ambiental motivat per les seves aplicacions en l'assistència sanitària, les llars intel·ligents o la planificació urbana. No obstant això, la majoria dels sistemes utilitzats per a aquestes aplicacions són *black-boxes* basades en el *deep learning* i, per tant, no poden ser inspeccionades, de manera que la raó de les seves decisions és confusa. Malgrat els avenços recents, encara hi ha una manca d'investigació en el *deep learning* interpretable en el domini d'àudio. Aquesta tesi té com a objectiu reduir aquest buit proposant diversos models de *deep learning* per a la classificació automàtica del so i la detecció d'esdeveniments.

Comencem descrivint una eina de programari de codi obert per a la investigació reproduïble en el camp del reconeixement de so, que es va utilitzar per implementar els models i executar experiments presentats en aquest document. A continuació, proposem un *front-end* interpretable basat en el coneixement del domini per adaptar les capes d'extracció de característiques d'una xarxa d'extrem a extrem per a la detecció d'esdeveniments sonors. Llavors presentem un nou model interpretable de *deep learning* per a la classificació automàtica del so, que explica les seves prediccions basades en la similitud de l'entrada a un conjunt de prototips apresos en un espai latent. Aprofitem el coneixement del domini dissenyant una mesura de similitud dependent de la freqüència. El model proposat aconsegueix resultats comparables als mètodes més moderns. A més, presentem dos mètodes automàtics per a reduir el model proposat que explota la seva interpretabilitat. Aquest model està acompanyat per una aplicació web per a l'edició manual del model, que permet una formulació de depuració *human-in-the-loop*. Finalment, proposem una extensió d'aquest model que funcioni per a un entorn polifònic, com la tasca de detecció d'esdeveniments sonors. Per proporcionar interpretabilitat, aprofitem l'formulació de la xarxa prototip i els mecanismes d'atenció.

Les eines per a la investigació reproduïble i els models interpretables de *deep-learning*, com els proposats en aquesta tesi, poden contribuir al desenvolupament d'una intel·ligència artificial més responsable i fiable en l'àmbit de l'àudio.

Contents

List of figures

List of tables

Glossary

ER Error Rate. xxi, 27, 61, 62, 91, 92

$F1$ F-score. xxi, 27, 59, 61, 62, 91, 92

FN False Negatives. 27

FP False Positives. 27

TP True Positives. 27

AGC Adaptive Gain Control. 18

AI Artificial Intelligence. 1, 9, 32, 39

APNet Audio Prototype Network. xix, xxi, 12, 33, 36, 66, 68, 72, 74, 75, 76, 80, 83, 84, 85, 87, 88, 93, 99, 100, 102, 103

BCE Binary Cross-Entropy. 22

CAM Class Activation Maps. 29

CAV Concept Activation Vector. 28, 31, 35

CCE Categorical Cross-Entropy. 21

CNN Convolutional Neural Network. xv, 17, 22, 23, 24, 28, 37, 54, 58, 74, 91, 99, 100, 101, 102, 103

CQT Constant-Q transform. 19, 20, 22

CRNN Convolutional Recurrent Neural Network. xvii, 17, 25, 74, 100

CW Concept Whitening. 28, 31

DCASE Detection and Classification of Acoustic Scenes and Events. xvii, 3, 4, 9, 10, 20, 24, 26, 47, 51, 97, 98

DCT Discrete Cosine Transform. 18

DNN Deep Neural Network. 1, 5, 6, 18, 65, 101

DRC Dynamic Range Compression. 18, 61

ERB equivalent rectangular bandwidth. 19, 20

ESR Environmental Sound Recognition. 2, 3, 11, 12, 20, 25, 100

GDPR European Union's General Data Protection Regulation. 1

GFCC Gammatone feature cepstral coefficients. 18

GMM Gaussian mixture models. 18, 20

GPU Graphics Processing Unit. 4

GRU Gated Recurrent Unit. 24

HMM Hidden Markov models. 20

LIME Local interpretable model-agnostic explanation. 6, 29, 34, 35, 99

LRP Layer-wise Relevance Propagation. 29

LSTM Long Short-Term Memory. 24, 74

MFCC Mel–Frequency Cepstral Coefficient. 17, 18, 20, 44

MLP Multi-layer Perceptron. xvii, 21, 22, 23, 74, 91, 102

NMF Non-negative Matrix Factorization. 35

PCEN Per-channel energy normalization. xviii, 12, 18, 19, 54, 57, 59, 60, 61, 62, 63

RNN Recurrent Neural Network. xvii, 17, 24, 25

SED Sound Event Detection. 3, 12, 16, 17, 18, 21, 22, 23, 25, 26, 27, 33, 37, 54, 56

SGD stochastic gradient descent. 22

STFT Short-time Fourier Transform. 16, 17, 44

SVM Support vector machines. 20

CHAPTER 1

Introduction

1.1 Motivation

The popularization of deep learning has led to significant advances in a wide range of scientific fields and practical problems (LeCun et al., 2015; Goodfellow et al., 2016), most notably in computer vision (Krizhevsky et al., 2012) and natural language processing (Peters et al., 2018), but also in the audio domain, improving the state of the art of several tasks, such as speech recognition (Hinton et al., 2012) and music recommendation (Schedl, 2019). Despite the recent progress, it is often hard to provide insights into the decision-making process of deep neural networks (DNNs). Their deep recursive structure and non-linear transformations allow DNNs to learn relevant representations of the data but, simultaneously, make it difficult to trace which aspects of the input drive their decisions. This black-box nature of DNNs may lead to unintended effects, such as reinforcing inequality and bias (Zhang et al., 2018; Chen et al., 2019b; Menon et al., 2020), and makes it difficult to extract the knowledge about the problem that the model captured in a way that humans can understand (Molnar, 2019). Particularly, in applications that impact human lives—such as in healthcare—the lack of transparency and accountability can have serious consequences (Rudin, 2019). It can be argued that the difficulty of understanding these models is not problematic in many successful applications of DNNs (Ras et al., 2022; Krishnan, 2020). However, high-probability prediction does not guarantee that the model always behaves as expected. For instance, it has been shown in image recognition that DNNs can be fooled with certain ease in classification tasks by adversarial attacks (Szegedy et al., 2014)—see (Carlini & Wagner, 2018; Qin et al., 2019) for examples in speech.

Integrating such algorithms into our daily lives requires wide social acceptance, but the unforeseen malfunctioning and side effects mentioned above undermine trustworthiness. Such concerns emerge in the new artificial intelligence (AI) regulations, like the legal notion of a *right to explanation* in the European Union's General Data Protection Regulation (GDPR) (Goodman & Flaxman, 2017). In consonance with this, there is a recent surge of research on machine learning models that provide explanations of their decisions in some level of detail, a field that is commonly known as *interpretable machine learning* (Gilpin et al., 2018; Molnar, 2019). Moreover, models that can be interpreted can be better debugged and audited in order to devise defense methods, foresee malfunctions, discover edge cases, or detect biases (Zhang & Zhu, 2018; Molnar, 2019).

In recent years, there has been an interest in developing technologies for Environmental Sound Recognition (ESR). These technologies are used for different applications, for instance, related to identifying sound sources in urban (Bello et al., 2018) and residential areas (Mesaros et al., 2016b) for noise mitigation, healthcare applications (Drugman et al., 2012), bioacoustic monitoring systems (Stowell, 2018; Cramer et al., 2020), assistance to people affected in their hearing capabilities (Hüwel et al., 2020; Messner et al., 2020), acoustics surveillance for security (Crocco et al., 2016; Sánchez-Hevia et al., 2017), or systems for industrial applications (Morrison & Pardo, 2019). Most of these systems are based on the application of signal processing and machine learning techniques to address the automatic generation of high–level descriptions of the environment, including detecting sound sources. However, most of the systems used for these applications are deep-learning-based black boxes and therefore can not be inspected, so the rationale behind their decisions is obscure. Notwithstanding some recent advances, there is still a lack of research in interpretable machine learning in the audio domain.

In this thesis, we propose interpretable machine learning methods for sound event detection and classification. The main focus is on environmental sound recognition since that, under real-world conditions, it is a challenging case of study with very relevant applications. However, the proposed approaches can be applied to other audio-related problems, such as music-related tasks.

1.2 Environmental sound recognition

The Environmental Sound Recognition field aims to study computational methods for the automatic analysis and understanding of environmental sound in different contexts. For example, sound event detection and classification tasks aim to identify the sounds occurring in our daily lives and assign a label within a target set of sound classes. In recent years, there has been an increasing interest in developing technologies for monitoring and diagnosing urban sound environments, which can facilitate the planning and management of the city to control noise pollution (Bello et al., 2018; Daniel Steele & Guastavino, 2013). The proposed systems are usually based on distributed sensor networks over the Internet of Things (IoT) technologies that provide sound pressure level estimates throughout the city in real-time (Bello et al., 2018; Mydlarz et al., 2015, 2014). Furthermore, based on the application of signal processing and machine learning techniques, some recent works (Bello et al., 2018; Salamon et al., 2014; Salamon & Bello, 2015b) have addressed the automatic generation of high–level descriptions of the sound environment, including the detection of sound sources. This information can help city councils to implement corrective policies or develop monitoring plans.

Environmental sound recognition can be tackled in different ways (Virtanen et al., 2018). A possible approach, called sound classification, aims to identify the predominant sound source at each audio signal time. A more complex approach, called Sound Event Detection (SED), is defined as the task of finding individual sound events by indicating the onset time, the duration, and a text label describing the type of sound.

Under real conditions, for instance, in urban environments, these problems can be very challenging. Moreover, the acoustic features of each class can have great diversity, given by the intrinsic variability of sound sources of the same type (e.g., cars) and the influence of the acoustic environment (e.g., reverberation, distance). Besides, the temporal overlapping of the sound events makes the classification task harder. There are other issues, such as the influence of the microphone's response (Virtanen et al., 2018).

The Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge and Workshop (Mesaros et al., 2018a) are annual venues that promote research in ESR tasks, including evaluating algorithms in standard publicly available datasets. The number of participants and authors has risen in the last years, illustrating this research area's relevance. In the context of this thesis, we have collaborated to this research community by developing software tools for reproducibility (Chapter 3 of this thesis and Zinemanas et al. (2020), Fuentes et al. (2021)), publishing datasets (Zinemanas et al., 2019b; Fuentes et al., 2022) and proposing novel interpretable methods (Chapter 5 of this thesis and Zinemanas et al. (2021b)).

1.3 Paradigm shift

During the past few years, advances in computing power, optimization algorithms, and available data have led to the generalized use of deep-learning models for audio-related problems. This process is usually known as a paradigm shift in the sense of Kuhn (1970) "The structure of scientific revolutions." In machine learning, Khun's paradigms can be associated with the model types, for instance, decision trees, support vector machines, or deep-learning models (Dotan & Milli, 2020; Pelillo & Scantamburlo, 2013). Machine learning researchers committed to a paradigm focus on increasing the scope and precision of that model type. Based on Kuhn's paradigms, Dotan & Milli (2020) argue that model types guide the research of the community members committed to them. Moreover, as happened in the last few years, when many people commit to the same model type, the discipline changes; there is a paradigm shift.

The number of papers referring to deep-learning models within a community can illustrate this paradigm shift. For instance, Figure 1.1 depicts the percentage of deep-learning related papers in the proceedings of the DCASE workshop over the years, showing that these models have become ubiquitous. The mass commitment to deep learning reinforces the predominance of this model type. For example, the deep-learning paradigm shift has promoted the availability of data, computing power, and the development of GPUs, which are pre-requisites for deep learning, reinforcing the paradigm (Dotan & Milli, 2020).

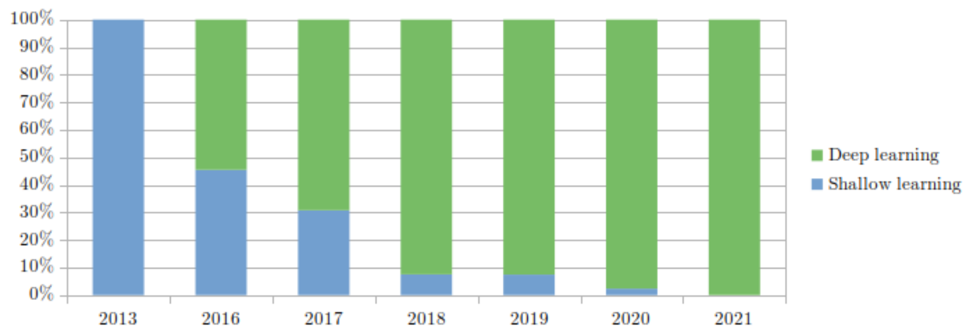


Figure 1.1: Percentage of papers referring to shallow and deep-learning models within the DCASE Workshop Proceedings over the last few years. For the calculation, we do not consider papers non-related to machine learning models, such as presenting datasets, challenges or software tools.

However, this paradigm shift is also value-laden because it implicates political and social values. For instance, the dependency on massive datasets and expensive hardware (like GPUs and powerful computers) favor the concentration of power of big companies in rich countries. Besides, the most used evaluation criteria for comparing models, such as accuracy or precision, are also value-laden since possible social or racial biases are not considered (Dotan & Milli, 2020). Deep-learning models trained on unbalanced datasets usually represent better the majority group achieving lower performances for the minority groups. For instance, this issue has been studied in tasks such as facial recognition (Xu et al., 2020) and criminal justice (Tolan et al., 2019). Other studies investigate different types of bias in Automatic Speech Recognition systems in English (Koenecke et al., 2020), French (Garnerin et al., 2019), and Arabic (Abushariah & Sawalha, 2013).

On the other hand, interpretable models can be used or devised to find and mitigate biases in deep-learning systems (Lee et al., 2019; David et al., 2020). Therefore the development of interpretable models in the audio domain can be helpful to mitigate the problems of deep-learning systems, such as algorithmic biases.

1.4 Interpretable Machine Learning

Several terms, such as *interpretability*, *explainability*, *intelligibility*, *comprehensibility*, and *transparency*, are used in this field to broadly refer to the capability of understanding how the model works, i.e. the process that led to its output (Lipton, 2016). Naturally, this does not mean a low-level mechanistic understanding of the model's inner workings but a rationale behind its predictions in a way that the user can interpret (Gilpin et al., 2018). The lack of agreement on the terms is misleading, and it mainly arises from their lack of precise meanings when applied to algorithms (Krishnan, 2020).

A problematic aspect of interpretability is that it is a domain-specific notion; therefore, there is no single definition of what attributes render models interpretable (Rudin, 2019). Interpretability is often materialized into a set of application-specific constraints on the model, dictated by domain knowledge, such as causality, monotonicity, or sparsity. In this thesis, we apply domain knowledge to extract more interpretable audio representations in end-end deep-learning models (Chapter 4). We also propose intrinsic interpretable models for classifying and detecting events based on prototypes (Chapter 5) and attention maps based on local prototypes (Chapter 6). These interpretable models should be able to produce explanations in terms of sound prototypes.

Another critical issue is that interpretability can be accomplished at different degrees, from a fully transparent to a mildly constrained model (Rudin, 2019; Gilpin et al., 2018). The design of fully-transparent machine learning models is very challenging, so we instead include constraints and layers based on domain knowledge to bring some interpretability to the deep-learning models.

In light of the above, it is not surprising that there are a lot of different approaches to explaining DNNs (Guidotti et al., 2018; Ras et al., 2022; Gilpin et al., 2018; Molnar, 2019). Most of them focus on explaining how the data are processed or represented by the network, namely a post-hoc explanation, as it pertains to an existing opaque model. Another approach is to create architectures designed to produce explanations or facilitate the interpretation of the network behavior, namely explanation-producing or inherently/intrinsically interpretable models.

The challenge in intrinsically interpretable machine learning is to create models that fit the data accurately while uncovering the types of patterns that the user would find interpretable. However, there is a widespread belief that interpretability and accuracy stand in conflict, assuming that the most accurate models must be inherently complex and, hence, uninterpretable (Gilpin et al., 2018). However, some evidence suggests that such a trade-off does not necessarily hold since, for instance, certain forms of interpretability have been integrated into deep neural networks for computer vision without losing accuracy (Li et al., 2018a; Chen et al., 2019a). On the contrary, the ability to interpret the model can reveal the weaknesses of the data or the processing, which can lead to performance improvements if adequately addressed.

1.4.1 Post-hoc explanations

Concerning post-hoc interpretability, the fundamental challenges in explaining large neural networks are the high number of parameters and the stacked non-linear operations involved. This can be tackled by training a linear proxy model that imitates the input-output behavior of the original opaque model, but it is easier to interpret (a white box). One of the classical methods of this kind is the local interpretable model-agnostic explanations (LIME) (Ribeiro et al., 2016), which learns a linear model as a local approximation of the black-box model in the neighboring space of an input data point. Other proxy models attempt to replicate the behavior across the entire dataset while using decision trees or rule extraction techniques (Gilpin et al., 2018). However, a proxy model is a linear approximation of a non-linear model and may be an inaccurate representation in some parts of the input space. Thus, the proxy models are often not reliable (Rudin, 2019).

Another type of post-hoc explanation focuses on studying the representation of the data inside the network. These methods try to explain the role of the layers (Yosinski et al., 2014), individual units (Zhang & Zhu, 2018), or other helpful representations (Kim et al., 2018). Besides, some visualization methods highlight the input characteristics that strongly influence the output (Ras et al., 2022). Among the most common ones are the *saliency maps*, which identify input features that cause a maximum response or portions where changes would most affect the output (Simonyan et al., 2014). They can be helpful to reverse-engineer importance values or sensitivities of inputs but do not disclose how this relevant information is being used so that they can produce unreliable explanations (Adebayo et al., 2018).

1.4.2 Intrinsic interpretability

For already trained models, post-hoc explanations may be the only option. However, an inherently interpretable model provides explanations faithful to what the model actually computes (Rudin, 2019). Several different approaches may be taken for designing interpretable or explanation-producing neural networks (Gilpin et al., 2018). For instance, DNNs can be trained to learn disentangled latent representations (Narayanaswamy et al., 2017) explicitly or to create generative explanations. Besides, attention mechanisms that proved very effective in natural language processing and computer vision provide a distribution over input units indicating their relative importance, which can be considered a form of explanation. However, more research is needed to assess whether attention mechanisms can provide reliable explanations (Wiegreffe & Pinter, 2019).

Recent approaches for designing interpretable neural network architectures based on explanations through prototypes and concepts, which have been mainly applied to the classification of images, are of particular relevance to our work (Li et al., 2018a; Alvarez Melis & Jaakkola, 2018; Hase et al., 2019; Chen et al., 2019a, 2020). Prototype classification is a classical form of case-based reasoning. Hence, it is an appropriate approach for interpretability grounded on how domain experts explain the reasoning processes behind complicated classification tasks in the visual domain. The architecture proposed in (Li et al., 2018a) appends a particular prototype layer to the network. Those prototypes are learned during training, and the network's predictions are based on their similarity to input in the latent space. Thus, the explanations that are given by the network are the prototypes and corresponding distances. Subsequently, in (Hase et al., 2019), this is extended to use hierarchically organized prototypes to classify objects in a predefined taxonomy, and in (Chen et al., 2019a), to learn parts of the training images as prototypes of each class. Finally, the approaches in (Alvarez Melis & Jaakkola, 2018; Chen et al., 2020) can be regarded as generalizations beyond similarities to prototypes into more general interpretable concepts.

1.5 Interpretability of audio-based models

Deep-learning based systems for audio classification can be illustrated as in Figure 1.2. Signal processing methods are used to extract an input representation from the audio waveform by computing some perceptually-motivated hand-crafted features that are inherently interpretable (such as energy distribution across different frequency bands). Then, a deep-learning model maps the input space into the output space by predicting one of the classes.

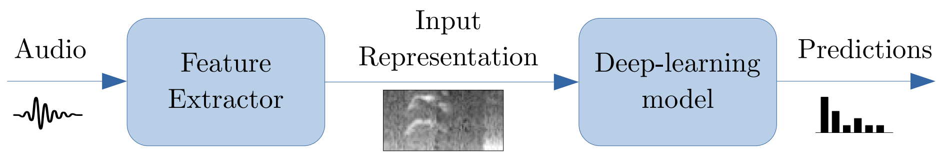


Figure 1.2: Diagram of a deep-learning-based system. First stage is devised to extract an input representation from the audio signal. Then the deep-learning model makes the prediction based on this input representation.

As mentioned in the previous section, post-hoc methods can be used to interpret the functioning of the black boxes or to explain local predictions. However, in this thesis, we focus on designing intrinsic interpretable models for classification and sound event detection. We include domain knowledge in the design process to make the models more interpretable and accurate. In Chapter 5, we propose an interpretable deep-learning model for sound classification based on prototypes. We show that this model can be both interpretable and accurate for environmental sound, music instrument, and speech classification. However, the characteristics of this model make it unsuitable for a polyphonic setting, such as the sound event detection task, so we propose an extension for this problem using local prototypes and attention maps in Chapter 6,

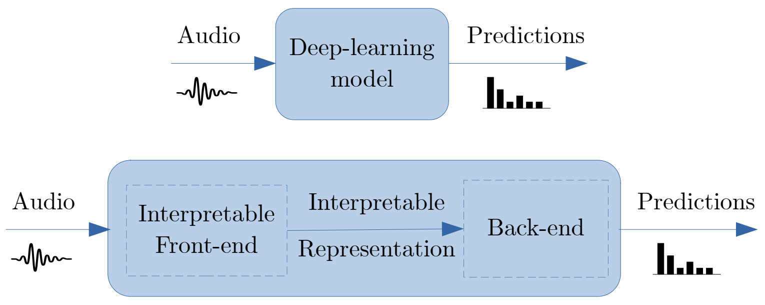


Figure 1.3: Diagram of a end-to-end deep-learning based system (top). In this case, the predictions are made directly from de waveform. First layers (front-end) of the model can be devised to extract interpretable features (bottom).

Alternatively, end-to-end models can be trained to obtain the predictions directly from the waveform signal (See Figure 1.3). In this case, the first layers of the deep-learning model are intended to do the feature extraction. For example, it has been shown that the first layers of end-to-end convolutional neural networks that learn representations from raw audio data extract features similar to the spectrogram or energies in mel-frequency bands (Thickstun et al., 2017; Lee et al., 2018; Tax et al., 2017). Additionally, some works have addressed the design of the first layers of these networks to tailor interpretable feature extraction stage using parametric filters (Loweimi et al., 2019; Pan et al., 2020; Won et al., 2020). In Chapter 4 we propose to include trainable hand-crafted kernels in the first layers of the network and use parametric normalization to make end-to-end networks more interpretable.

1.6 Reproducibility and interpretability

Another side effect of the deep-learning paradigm shift is that most modern signal processing methods and models are considerably more complex than a decade ago and heavily rely on the data and software used for their implementation (McFee et al., 2019). Besides, implementation details can profoundly affect the reported performance of a method (Vandewalle et al., 2009; Raffel et al., 2014; Six et al., 2018). Therefore, it has become increasingly challenging to reproduce the findings or to compare a new method to earlier ones based only on the description of research systems found in publications (McFee et al., 2019; Six et al., 2018). This can slow down progress in the research community due to the effort devoted to reimplementing methods and dealing with all details of different software packages and datasets. For the optimal reuse of scientific research, numerous authors and institutions are advocating not only for open-access publications and data but also for the release of software and models that can be of use across a variety of domains (Colom et al., 2015; McFee et al., 2019; Bjornson, 2019). This poses several challenges, such as the development of best practices guidelines to stick to, the need for long-term funds for those initiatives, and the coordination of research efforts–which is crucial to the quality of research (Leonelli & Ankeny, 2015).

Fortunately, some scientific communities, such as DCASE, are increasingly endorsing transparency and best practices for reproducible research. The DCASE Challenge provides an annual benchmark of methods for different tasks (Mesaros et al., 2018a, 2019), and since 2019 it has given specific awards for open–source and reproducible methods. In addition, there are numerous readily available datasets (Piczak, 2015; Salamon et al., 2014; Mesaros et al., 2017, 2018c; Fonseca et al., 2018; Cartwright et al., 2019; Zinemanas et al., 2019b; Heittola et al., 2020) and various authors publicly release software tools as well as easily usable and well-documented implementations of their methods (Salamon et al., 2017; Salamon & Bello, 2017; Mesaros et al., 2016a; Bilen et al., 2020).

While all such resources are of great value for the progress of the field, there are still several opportunities to improve research reproducibility. For instance, usability aspects have to be considered (e.g., documentation and installability) so that a piece of code is functional rather than simply available. In addition, when trying to compare the results of a new method with earlier methods, it is often the case that researchers have to reimplement those baselines or manage to find and modify an existing implementation. This is time-consuming and implies dealing with various coding conventions and software tools. With the intrinsic complexity of deep–learning methods, delving into a given DCASE problem can be tricky, especially for newcomers.

Furthermore, both interpretable models and reproducible experiments are crucial to fostering a more responsible and trustworthy AI (High Level Expert Group on Artificial Intelligence, 2019; Barredo Arrieta et al., 2020). Therefore, a framework for the reproducible development and use of such systems would be beneficial in the context of interpretable deep learning for audio-related problems.

In the light of the above, in the context of this thesis, we have developed

DCASE-models[1], an open-source Python library whose main goal is to facilitate the development of deep-learning models for DCASE-related tasks such those tackled in this dissertation (Chapter 3). All deep-learning methods used in this thesis (Chapters 4, 5 and 6) are implemented using this library.

1.7 Scope and goals

This thesis focuses on the design of interpretable models for environmental sound classification and detection. We use domain knowledge to design parts or layers of the networks that are intrinsically interpretable. In Chapter 4 we show how we devise the first layers of an end-to-end deep-learning model to extract meaningful feature representations. In Chapter 5 we propose a prototype-based neural network for sound classification. In Chapter 6 we extend this prototype-based model by adding an attention mechanism to work in a polyphonic setting such as the sound event detection task. We also show that reproducible research is fundamental for fostering interpretable models, and this thesis presents a Python library for reproducible development of deep-learning models for environmental sound recognition (Chapter 3).

Since there is a lack of research on interpretable deep-learning models in the audio domain, this thesis aims to promote this type of model in these research communities. Furthermore, given that this thesis focuses on environmental sound recognition, we mainly contribute to the DCASE community. Therefore, the specific goals of this thesis are:

1. To develop an open-source software library for the reproducible use and development of machine learning models in the context of the DCASE community.
2. To design interpretable deep-learning representations that are suitable for different audio-related problems.
3. To devise intrinsic interpretable deep-learning models for classifying sources and detecting events in the context of environmental sounds.
4. To contribute to the research community in curating data and developing software tools to ease the use of publicly available datasets.

Objectives 1 to 3 are covered in this document, as explained in the next section. Objective 4 includes other contributions and collaborations not included in this document. In the context of the DCASE community, we also contributed to developing a dataset for sound event detection in urban environments (Zinemanas et al., 2019b). We have also collaborated to curate a dataset for

[1]https://github.com/MTG/DCASE-models

audio-visual scene understanding, specially devised for training self-supervised models (Fuentes et al., 2022). Another relevant collaboration is the development of soundata[2], an open-source Python library for downloading, validating, and loading common sound datasets (Fuentes et al., 2021).

1.8 Outline of the thesis

In this section, we describe the structure of the thesis and provide a brief description of each chapter, including the main goals and the achievements reported. Figure 1.4 shows a diagram of outline of the thesis in relation with the main topics and the specific goals defined previously.

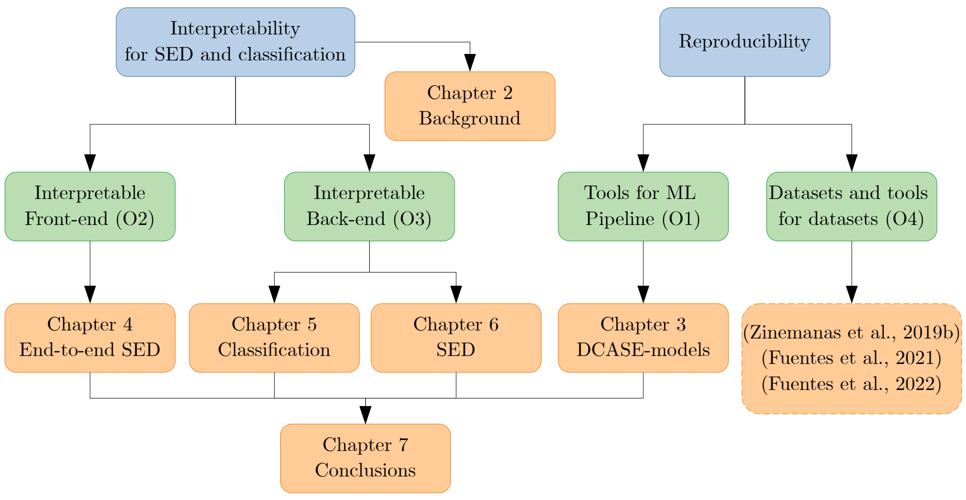

Interpretability for SED and classification
Reproducibility
Chapter 2 Background
Interpretable Front-end (O2)
Interpretable Back-end (O3)
Tools for ML Pipeline (O1)
Datasets and tools for datasets (O4)
Chapter 4 End-to-end SED
Chapter 5 Classification
Chapter 6 SED
Chapter 3 DCASE-models
(Zinemanas et al., 2019b) (Fuentes et al., 2021) (Fuentes et al., 2022)
Chapter 7 Conclusions

Figure 1.4: Diagram of the thesis outline in relation with the topics and the specific objectives defined in Section 1.7. We also include contributions outside this thesis.

Chapter 2 summarizes the background to support the topics covered in this thesis. It includes a literature review of the fields involved in this thesis. We first define the tasks of environmental sound recognition tackled in the dissertation. We then explain the most commonly used input representations for deep-learning-based ESR systems. We include a description of the most relevant deep learning models and a summary of the available datasets and the evaluation metrics we used. We follow the chapter with a comprehensive literature review of the field of interpretable deep learning. We review both post-hoc and intrinsic interpretable methods. We finish the chapter with a literature review of interpretable models used for audio-related tasks.

In Chapter 3 we describe DCASE-models, an open-source Python library devised to alleviate the development of deep-learning-based systems for ESR. We first explain the design principles and the practices we followed to devise the

[2]`https://github.com/soundata/soundata`

library. We then describe how the library is organized to include all steps in the typical pipeline of an ESR system. Finally, we finish the chapter by showing a few examples of how using the library facilitates the development of such models. This chapter is mainly based on the following publication:

- Zinemanas, P., Hounie, I., Cancela, P., Font, F., Rocamora, M., Serra X. DCASE-models: A Python library for computational environmental sound analysis using deep-learning models. *5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2020)*, November 2020.

In Chapter 4 we propose an end-to-end deep-learning architecture for SED whose first layers are devised to extract meaningful features from the audio signal. We first describe how we calculate the mel-spectrogram in the model and compare it with a previously proposed method. We then show how we adapt PCEN, a parametric per-channel normalizer, to scale the input representations. We finish the chapter with experiments and the results obtained, including a comparison with the baseline systems. Finally, we show how we use the insights from the interpretable layers to reduce the number of parameters. The contributions of this chapter were published in:

- Zinemanas, P., Cancela, P., Rocamora, M. End-to-end convolutional neural networks for sound event detection in urban environments. *24th Conference of Open Innovations Association (FRUCT 2019)*, March 2019.

Chapter 5 presents an interpretable deep learning model for sound classification based on prototypes, namely APNet. We first define the main contributions of this chapter and present the proposed model. We then describe the datasets, baselines, and training strategy used in this work. We also present the experiments and the results obtained. We show that the model is successful in addressing different audio classification tasks, including speech, music, and environmental sound. We also include the analysis of the interpretable parameters, including the learned prototypes. We then show how we use the insights of the interpretable layers to refine and prune the network. Finally, we finish the chapter by discussing the main conclusions. The content of this chapter is based on the following journal paper:

- Zinemanas, P., Rocamora, M., Miron, M., Font, F., Serra, X. An Interpretable Deep Learning Model for Automatic Sound Classification. *Electronics 10*(7:850) 2021, `https://doi.org/10.3390/electronics10070850`

In Chapter 6 we discuss the limitations of APNet for representing overlapping sound sources in the latent space, like in the sound event detection task. Therefore, we propose a solution based on attention maps and local prototypes. We first define the proposed model and the objective function used to optimize the model. We then show the experiments and the results, including an analysis of the learned prototypes and the explanations the model provides based on the attention maps. This chapter is based on the following publication:

- Zinemanas, P., Rocamora, M., Fonseca, E., Font, F., Serra X. Toward interpretable polyphonic sound event detection with attention maps based on local prototypes. *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2021)*, November 2021.

We include a summary and discussion of the key results and takeaways at the end of each chapter. Additionally, we finish this thesis with Chapter 7 by presenting an overall summary and the conclusions. We also discuss future perspectives to develop the interpretable deep learning field within the audio research community.

CHAPTER 2

Background

2.1 Introduction

This chapter provides the description and relevant background about the topics covered in this thesis, along with the related literature. Specifically, in Section 2.2, we further introduce the problem of environment sound recognition. We define the main tasks involved in this problem, the most popular input representations, and how those are used in the shallow learning paradigm context. We discuss the fundamentals of the deep-learning models applied to the sound event detection task and the most relevant works using these models. We also review the datasets for environmental sound recognition and present the evaluation metrics used in this thesis. Then, in Section 2.3, we survey the literature on interpretable deep learning including post-hoc and intrinsic methods. Finally, Section 2.4 presents the literature review of interpretable models used in the context of audio-related problems. Altogether, these literature reviews provide the related work for Chapters 4, 5, and 6. Throughout this chapter, we also define the scope of this thesis and motivate some of the choices made.

2.2 Environmental sound recognition

Environmental sound recognition involves machine listening systems that generate high-level semantic descriptors. These descriptors can be in different forms. We commonly use the following taxonomy to categorize them: classification, tagging, and sound event detection. Figure 2.1 illustrates the output obtained in each of these systems. Environmental sound classification is the task of classifying the predominant source present in an audio excerpt. Audio tagging refers to the multi-label classification of an audio snippet. A tagging system can provide all sound sources in an audio file and other high-level descriptors, such as the acoustic scene. A more complex approach, commonly known as Sound Event Detection (SED), aims to detect all sound sources in the audio file, including the onset time and the duration of each of them.

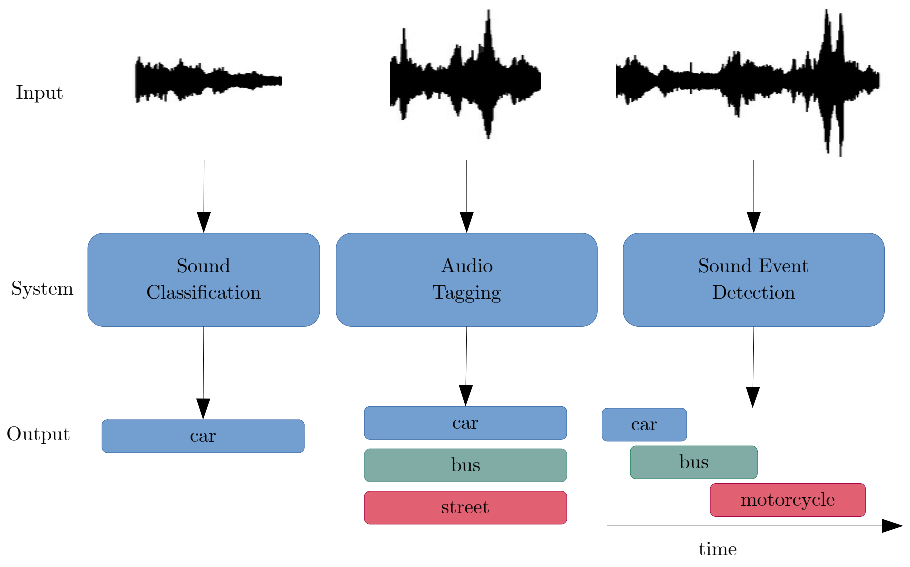


Figure 2.1: Diagrams of the three different approaches of environmental sound recognition: sound classification, audio tagging and sound event detection.

2.2.1 Input representations

Spectrograms

Usually, audio signals are transformed into a time-frequency representation that better characterizes the variation of its frequency components over time compared to the audio waveform. The most direct way to do this is using the Short-time Fourier Transform (STFT). Let $x[n]$ be the audio signal in the time domain, the STFT for the frequency component f, and the frame t is calculated by:

$$X[t, f] = \sum_{m=0}^{N} w[m] x[th + m] \exp\left(\frac{-j2\pi fm}{N}\right), \tag{2.1}$$

where $w[m]$ is the analysis window, N is the window length, and h is the hop size between frames. The spectrogram is calculated as the magnitude of the STFT for each time-frequency point. Even though the spectrogram has been used for sound event detection, for instance, as input of recurrent (Zöhrer & Pernkopf, 2016) or convolutional neural networks (Adavanne & Virtanen, 2017), other perceptually motivated variants are more popular.

Mel-spectrograms

Mel-spectrograms are arguably the most popular representation used for audio-based classification tasks. This representation is based on a non-linear warping of the frequencies into the mel-frequency scale, which motivated by auditory perception. The mel frequency scale is defined as:

$$\mathrm{mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right). \tag{2.2}$$

The mapping to the mel-frequency scale is obtained by applying a bank of triangle filters whose center frequencies are distributed linearly in the mel scale and whose bandwiths are almost constant until 1000 Hz and constant in the mel scale for higher frequencies. The bandwidth of the filters and their logarithmic distribution are inspired in the human auditory perception.

$$E[i, l] = \sum_{k} |H_l[k] X[i, k]|^2,\ l = 0, \ldots, L - 1, \tag{2.3}$$

where $H_l[k]$ is the frequency response of the l-th filter, and L is the total number of filters. Again, this energy distribution is calculated for each frame of the signal.

Most works on SED use the energy on mel bands as input features, in conjunction with Recurrent Neural Networks (RNNs) (Adavanne et al., 2016; Vu & Wang, 2016; Gorin et al., 2016; Zhou, 2017), Convolutional Neural Networks (CNNs) (Salamon et al., 2017; Jeong et al., 2017) or Convolutional Recurrent Neural Networks (CRNNs) (Cakir et al., 2017; Adavanne & Virtanen, 2017).

Mel–Frequency Cepstral Coefficients (MFCC)

The Mel–Frequency Cepstral Coefficients (MFCCs) were originally devised to describe the spectral content of speech signals by decomposing them into an excitation (carrier signal) and a frequency response (modulation) (Davis & Mermelstein, 1980). Therefore speech signals are modeled by the signal generated in the glottis and a filter representing the effects of the vocal tract. The MFCCs represent the frequency response so those are useful to recognize speech and the speaker. These features have been used in music-related tasks such as singer identification (Kim & Whitman, 2002) or instrument recognition (Herrera-Boyer et al., 2006). MFCCs have also been used for SED as features, with Gaussian mixture models (GMM) (Mesaros et al., 2016b), decision trees (Elizalde et al., 2016; Salamon & Bello, 2015b,a) and DNNs (Kong et al., 2016; Wei et al., 2016; Wang et al., 2017a; Lu & Duan, 2017).

The MFCCs are defined as the Discrete Cosine Transforms (DCTs) of the energy in the mel bands:

$$\mathrm{MFCC}[i,m] = \sum_{l=0}^{L-1} \log(E[i,l]) \cos\left(\frac{2\pi}{L} lm\right),\ m = 0,\dots,M-1, \tag{2.4}$$

where M is the number of cepstral coefficients (Davis & Mermelstein, 1980).

Although MFCCs are the most common cepstral representations used for sound classification, other cepstral representations, such as the Gammatone feature cepstral coefficientss (GFCCs), have also been applied for sound scene analysis (Valero & Alias, 2012).

Energy normalization

Generally, the mel–spectrogram, $E[i,l]$, is converted into decibels as follows:

$$E_{dB}[i,l] = 10 \log_{10}(E[i,l]), \tag{2.5}$$

where i is the hop index, and l is the frequency channel (mel filter index). This normalization is widespread in the SED problem (Salamon et al., 2017; Cakir et al., 2017). A new approach, called Per-channel energy normalizations (PCENs), was recently proposed to increase the robustness to loudness variations on speech detection systems (Wang et al., 2017b). The static logarithmic function is replaced with a dynamic range compression (DRC) and adaptive gain control (AGC) with temporal integration. This integration is performed with a low-pass filter ϕ_T, in a temporal scale of T, as follows:

$$E_{PCEN}[i,l] = \left(\frac{E[i,l]}{(\epsilon + M[i,l])^{\alpha}} + \delta\right)^{r} - \delta^{r}, \tag{2.6}$$

where $M[i,l] = \left(E^{t} * \phi_T\right)[i,l]$, while α, ϵ, r and δ are positive constants and $*$ denotes convolution (Lostanlen et al., 2019). The low-pass filter is implemented as a first order IIR filter:

$$M[i,l] = (1-s)M[i-1,l] + sE[i,l], \tag{2.7}$$

where s is a smoothing coefficient (Wang et al., 2017b).

Figure 2.2 shows the comparison of the logarithmic scale versus the PCEN using the parameter values suggested in (Wang et al., 2017b) for an example audio file from the URBAN-SED dataset.

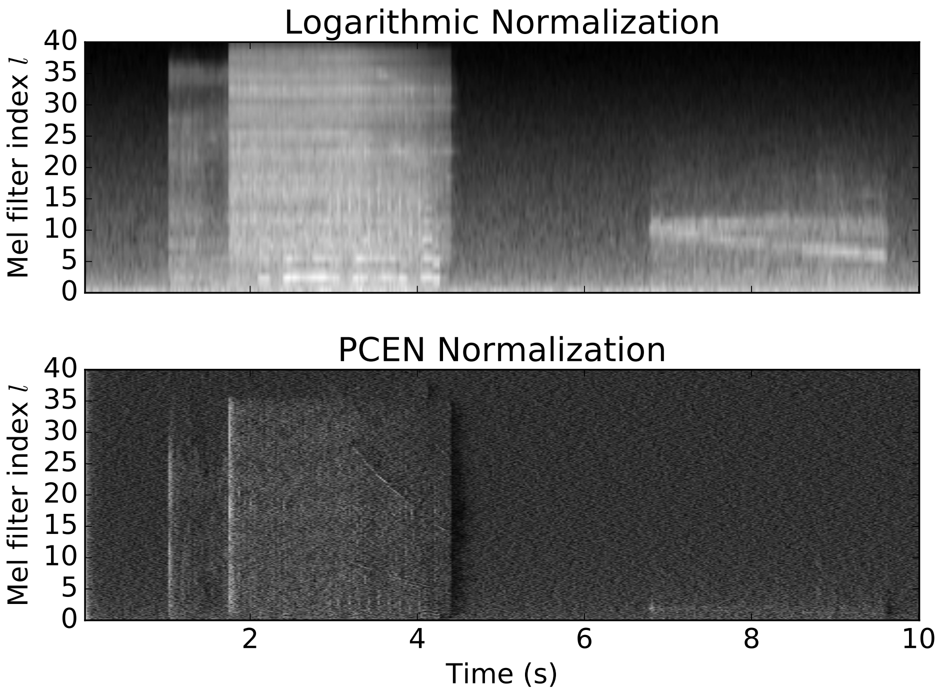


Figure 2.2: Comparison of energy normalization with logarithmic function (top) and using PCEN (bottom). PCEN is calculated using *librosa* implementation with parameters $\alpha = 0.8$, $\delta = 10$, $r = 0.25$, $T = 0.06$, $\epsilon = 10^{-6}$.

Values for the PCEN parameters have been proposed according to asymptotic studies (Lostanlen et al., 2019), but it is interesting to note that the function is differentiable; thus, those parameters could be learned by neural network models (Wang et al., 2017b). In Chapter 4, we explore this in more depth.

Other perceptually-motivated representations

Other perceptually-motivated representations for environmental sound recognition are based on the critical-band frequency mapping and the Constant-Q transform (CQT) (Brown, 1991). The former describes the bandwidth of the auditory filters in the cochlea (Fletcher, 1940). The equivalent rectangular bandwidth scale (ERB) estimates the frequency and bandwidth of these filters. The latter is a time-frequency representation where the frequencies are logarithmically spaced.

Multi-scale representations, such as wavelets, have also been used for sound recognition. Those functions integrate to zero and can form a basis by applying translation and dilation transformations.

Most of the representations presented above try to characterize the speech or music signals. These have been used for several tasks such as speech recognition, speaker identification, and instrument classification. However, features such MFCCs can capture global spectral envelope properties that are also useful for sound recognition tasks (Serizel et al., 2018). Some other works try to design special features for sound scene classification and event detection. For instance, Chu et al. (2009) proposed a matching pursuit-based method to obtain time-frequency representations. This method uses a dictionary for feature selection producing an interpretable set of features.

2.2.2 Shallow learning approach

The shallow-learning approach was the predominant paradigm before the irruption of deep learning In this approach, features extracted from the audio signal are the input of a shallow machine learning model that makes predictions to perform classification, tagging, or detection. A great deal of machine learning models have been applied for environmental sound recognition.

For instance, discriminative models, such as linear models or Support vector machines (SVM), try to predict the class from the input without explicitly modeling the input space (McFee, 2018). This type of models have been used for sound recognition tasks along with MFCCs (Vikaskumar et al., 2016), mel-spectrograms (Sena Mafra et al., 2016), CQT (Bisot et al., 2016) and Wavelets (Kun et al., 2017; Waldekar & Saha, 2018; Qian et al., 2017).

On the other hand, generative models, for instance, Gaussian mixture models (GMM) and Hidden Markov models (HMM), approximate the data generating process by modeling the joint distribution over the input and output spaces (McFee, 2018). For example, GMMs have been used for ESR with MFCCs (Mulimani & Koolagudi, 2016) and CQT (Abrol et al., 2017) as input representation. Moreover, HMMs were used to model temporal sequences using an ERB-based spectrogram as input (Benetos et al., 2016).

Moreover, decision-tree-based models, such as Random Forest, attempt to predict the output based on decision rules learned from the training data. These models have been used along with MFCCs (Salamon et al., 2014) for sound classification.

2.2.3 Deep-learning approach

As mentioned in Section 1.3, the DCASE community faced a paradigm shift from the design of handcrafted features to the deep learning approach. In this thesis, we focus on discriminative deep-learning models. These models include non-linear transformations over the input and a classifier to discriminate between classes, and both are fit simultaneously during the training process (McFee, 2018). In the following, we present some architectures and examples of these models used for SED and classification.

Multi-Layer Perceptron

The most simple neural network is the Multi-layer Perceptron (MLP) based on Rosenblatt's neuron model (Rosenblatt, 1958). It consists of a combination of intercalated affine transformations and non-linear activations. The affine function and the activation form one layer. Therefore for each layer i, let us define:

$$f_i\left(\boldsymbol{x}; \boldsymbol{W}_i, \boldsymbol{b}_i\right) = \rho_i\left(\boldsymbol{W}_i^T \boldsymbol{x} + \boldsymbol{b}_i\right) \tag{2.8}$$

where $\boldsymbol{x}$ is the input; $\boldsymbol{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$ and $\boldsymbol{b}_i \in \mathbb{R}^{d_i}$ are the layer's weights and bias respectively; and ρ_i is the activation function (Goodfellow et al., 2016; McFee, 2018). The dimension of each layer is d_i and each function f_i maps the space $\mathbb{R}^{d_{i-1}}$ into $\mathbb{R}^{d_i}$.

An MLP model includes several functions like f_i concatenated. Therefore, the output of the model for an input $\boldsymbol{x}$ is:

$$f\left(\boldsymbol{x}; \boldsymbol{\theta}\right) = \left(f_1 \circ f_2 \circ \cdots f_m\right)\left(\boldsymbol{x}\right) \tag{2.9}$$

where $\boldsymbol{\theta} = \left(\boldsymbol{W}_i, \boldsymbol{b}_i, \rho_i\right)_{i=1}^m$ is the set of parameters of the model.

The last layer's activation function must be chosen according to the task. For single-label multi-class tasks, such as sound classification, the softmax activation is commonly adopted, which forces all per-class output predictions to sum one. For multi-label classification tasks, such as sound event detection, the final activation is typically a sigmoid function per output channel, thus forming independent classifiers for each class. In this case, the labels are defined as a binary vector $y \in \{0, 1\}^C$, where C is the number of classes (McFee, 2018).

Training

The training of a neural network is performed by updating the network weights in order to minimize a loss function, $\mathcal{L}(\boldsymbol{x}, y; \theta)$, that expresses the divergence between the predictions $\hat{y} = f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)$ and the target labels y. The most commonly-used loss function for multi-class classification is the Categorical Cross-Entropy (CCE) loss:

$$\mathcal{L}_{CCE}(\hat{y}, y) = -\sum_{c=1}^{C} y_c \log \hat{y_c}, \tag{2.10}$$

where y_c and $\hat{y_c}$ are the c'th element of the target label and the predictions, respectively. Since y is a one-hot encoded vector, only one term of the summation contributes to the loss, and the rest of the classes are ignored. In multi-label classification, such as sound event detection, the network's output is composed of C independent binary classifiers. In this setting, binary classification loss functions are typically adopted. They consider positive examples and also negative ones. For example, BCE is the most commonly used loss function for this setting:

$$\mathcal{L}_{BCE}(\hat{y}, y) = \sum_{c=1}^{C} -y_c \log \hat{y_c} - (1 - y_c) \log(1 - \hat{y_c}). \tag{2.11}$$

Usually, the model parameters are fitted by a variation of the gradient descent method. This method is an iterative algorithm that can be described by:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\boldsymbol{x}, y; \theta^{(t)}) \tag{2.12}$$

where $\eta > 0$ is the learning rate and ∇_θ denotes the gradient with respect to parameters θ (Bishop, 2006; McFee, 2018). Since the $\mathcal{L}$ function involves the composition of a series of differentiable functions, its gradient can be calculated using the Chain Rule. This method is called error backpropagation since the errors in the last layers are backpropagated to the first layers (Bishop, 2006).

Stochastic methods are used to avoid calculating the loss function over the whole training set and to avoid local minima. Instead, the gradient is calculated over a random sample of the training set. This method is called stochastic gradient descent (SGD) (Bishop, 2006). Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), and Adam (Kingma & Ba, 2014) are other variants of optimizers that are adapted to the relative change of the parameters, reducing the dependency on the learning rate selection.

MLP for Sound Event Detection

MLP models have been used for SED and sound classification. Figure 2.3 shows the diagram of the MLP proposed by (Mesaros et al., 2017). The model's input is a vector of dimension $d_0 = 200$. This vector is formed by the logarithm of the energy in 40 mel bands from 0 to 22050 Hz of five frames of audio, including the central one, the two previous ones, and the two later ones. The analysis window is 40 ms, and the frames are overlapped by 50 %. The model includes two hidden layers of 50 units each.

Note that the network's predictions use a small amount of context information (120 ms). For non-stationary signals, the lack of context can be problematic. For instance, the two consecutive frames can be different. Besides, this type of architecture does not take advantage of the implicit structure of the audio signals in time or frequency (McFee, 2018). In the next section, we show that Convolutional Neural Networks can overcome this issue.

MLP models have been used for environmental sound recognition with other features as input, such as the spectrogram (Foleiss & Tavares, 2017) and the CQT (Bisot et al., 2017).

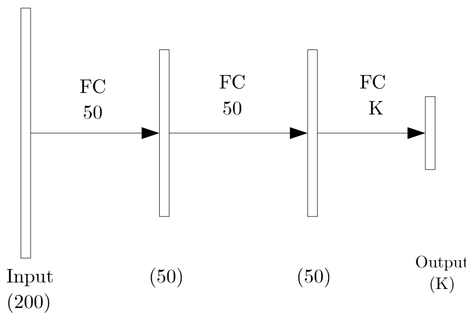


Figure 2.3: Diagram of the MLP model proposed by (Mesaros et al., 2017). It is formed by three fully-connected (FC) layers. It has two hidden layers of 50 units each and FC layer to map the predictions of K classes.

Convolutional Neural Network

Convolutional Neural Networks can be one-dimensional (for instance, waveform as input), two-dimensional (time-frequency representation as input), or of higher dimensions (video input or multiple time-frequency representations). In this thesis, we focus on two-dimensional CNN since those are the most used for the tasks of SED and classification.

The operation that defines one layer of convolution is the following:

$$f_i\left(\boldsymbol{X};\boldsymbol{W}_i,\boldsymbol{b}_i\right)=\rho_i\left(\boldsymbol{W}_i*\boldsymbol{X}+\boldsymbol{b}_i\right), \tag{2.13}$$

where $\boldsymbol{X}\in\mathbb{R}^{T_{i-1}\times F_{i-1}\times d_{i-1}}$ is the layer's input, $\boldsymbol{W}_i\in\mathbb{R}^{n_i\times p_i\times d_{i-1}\times d_i}$ is the layer's kernel and $\boldsymbol{b}_i\in\mathbb{R}^{d_i}$ is the bias. The output of the function f_i lies at the space $\mathbb{R}^{T_i\times F_i\times d_i}$. Note that T_i and F_i are the spatial dimensions (time and frequency respectively) of the input of layer i; d_i is the number of filters; and n_i and p_i are the dimensions of the convolutional kernel (Goodfellow et al., 2016; McFee, 2018).

If the model's input is a time-frequency representation, such as the spectrogram, the dimensions T_0 and F_0 are the number of frames and the frequency bins of the map respectively. The number of input channels is $d_0 = 1$ for monophonic inputs, $d_0 = 2$ for stereo signals, or $d_0 > 1$ if more complex representation are used, for instance the combination of several feature maps.

Typically, the convolution operations are used in the first layers of the network. Then, fully-connected (dense) layers, such as the ones from Equation (2.8), are used for the upper layers. The model proposed by Salamon & Bello (2017) is an example of this type of network. Figure 2.4 shows the diagram of this model. It has three convolutional layers and three fully-connected layers. The final layers has sigmoid activation over K classes to perform the multi-label classification.

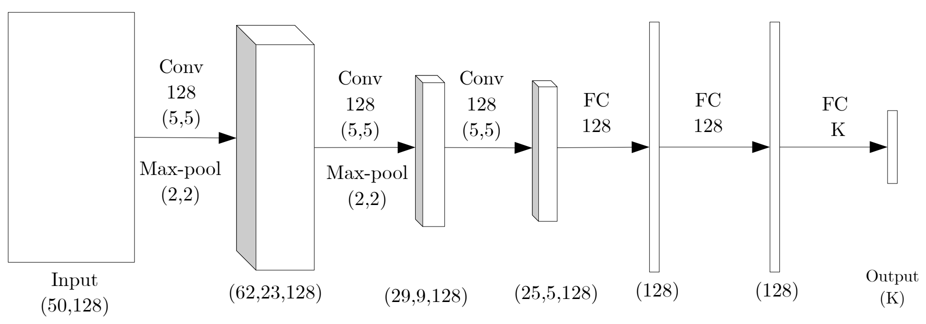


Figure 2.4: Diagram of the CNN model proposed by (Salamon & Bello, 2017).

Other CNNs have been used for environmental sound recognition with different types of inputs, such as mel-spectrogram (Sakashita & Aono, 2018; Dorfer et al., 2018), gammatone filters (Perez-Castanos et al., 2020), wavelets (Li et al., 2018b) and the audio waveform (Purohit & Agarwal, 2018; Pajusco et al., 2020).

Recurrent Neural Network

Recurrent Neural Network (RNN) integrate temporal information and are used to model sequential data. In the most simple recurrent layer, a state vector, $\boldsymbol{h}[t] \in \mathbb{R}^{d_{i-1}}$, is defined by:

$$\boldsymbol{h}[t] = \rho\left(\boldsymbol{W}^T\boldsymbol{x}[t] + \boldsymbol{V}^T\boldsymbol{h}[t-1] + \boldsymbol{b}\right), \tag{2.14}$$

where $\boldsymbol{x}[t] \in \mathbb{R}^{d_{i-1}}$ is the layer's input, $\boldsymbol{W} \in \mathbb{R}^{d_{i-1}\times d_i}$ is the weight matrix, $\boldsymbol{V} \in \mathbb{R}^{d_{i-1}\times d_i}$ is the recurrent weight matrix, and $\boldsymbol{b} \in \mathbb{R}^{d_i}$ is the bias (Goodfellow et al., 2016; McFee, 2018).

The layer's output is the sequence formed by the state vectors:

$$f_i\left(\boldsymbol{X}; \boldsymbol{W}_i, \boldsymbol{V}_i, \boldsymbol{b}_i\right) = \left(\boldsymbol{h}[t]\right)_{t=1}^{T}. \tag{2.15}$$

The function f_i maps the space $\mathbb{R}^{d_{i-1}}$ into $\mathbb{R}^{d_i}$. Normally, more complex recurrent layers are used, such as Gated Recurrent Unit (GRU) (Cho et al., 2014) or Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997).

The model proposed by Adavanne et al. (2016), which obtained the best result of the DCASE 2016 task 3 challenge (sound event detection in real life audio), is an example of this type of network. The input is the energy in 40 mel-frequency bands. The network has two hidden layers with 32 units each. Figure 2.5 shows the diagram of this model. Note that in this case, the predictions are made at the frame level, and the output has shape (T, K) instead of (K). Therefore, the predictions of the CNN at the frame level are aggregated to make a classification at a chunk of several frames; 1-second resolution in the case of the model proposed by (Salamon & Bello, 2017).

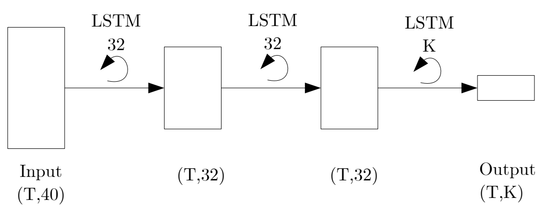


Figure 2.5: RNN proposed by (Adavanne et al., 2016)

Convolutional-Recurrent Neural Network (CRNN)

The model proposed by Cakir et al. (2017) is an example of how convolutional and recurrent layers can be combined. This model includes three convolutional layers, three recurrent layers, and one fully-connected layer. The convolution layers extract temporal features from the input integrated using the recurrent layers. Figure 2.6 shows the diagram of this model.

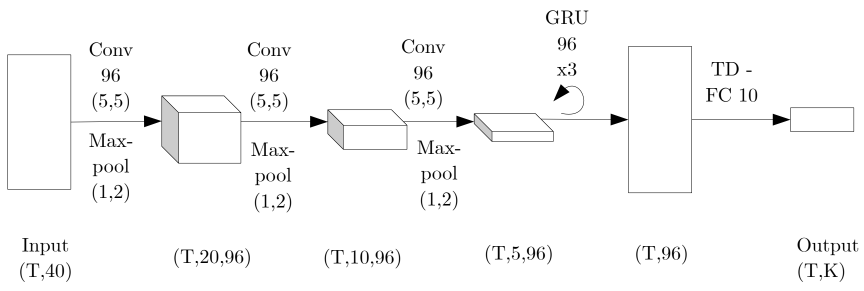


Figure 2.6: CRNN proposed by (Cakir et al., 2017)

Other CRNN models were proposed for SED and classification specially using mel-spectrograms as inputs (Ebbers & Häb-Umbach, 2019; Harb & Pernkopf, 2018; Lim et al., 2018, 2017).

2.2.4 Datasets

Publicly available datasets for ESR are of crucial importance to foster the development of the field as they encourage reproducible research and fair comparison of algorithms. In this respect, the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge, held for the first time in 2013 and repeated every year since 2016, has established a benchmark for sound event detection and acoustic scene classification using open data (Mesaros et al., 2018a, 2019).

Two of the datasets used in the DCASE challenge for SED in urban environments are part of the TUT database (TUT Sound Events 2016 and 2017), which was collected in residential areas in Finland by Tampere University of Technology (TUT) and contain overlapping sound events manually annotated (Mesaros et al., 2016b). The classes were defined during the labeling process. In a first step, the participants were asked to mark all the sound events freely, and later the labels are grouped into more general concepts. In addition, the tags must be composed of a noun and a verb, such as ENGINE ACCELERATING or BRAKES SQUEAKING (Mesaros et al., 2016b).

Datasets for SED usually include a limited amount of annotated audio, primarily due to the amount of time that annotating multiple overlapping sounds takes. For instance, TUT Sound Events 2017 dataset (Mesaros et al., 2017) is formed by two hours of recordings in residential areas and only includes seven classes. MAVD dataset (Zinemanas et al., 2019b) has four hours of annotated audio in urban environments focusing on traffic noise. On the other hand, the SINGA:PURA (Ooi et al., 2021) dataset has more than 18 hours of annotated audio following the taxonomy of urban sounds proposed in SONYC-UST (Cartwright et al., 2020).

Using weak labels that only indicate the presence of the source without time boundaries alleviates the work involved in the manual annotation. For example, FSD50K (Fonseca et al., 2022) and Audioset (Gemmeke et al., 2017) datasets have weak labels to describe the audio files. These are much bigger datasets and intend to be more general in terms of target classes. Audioset follows a hierarchical ontology of sound events formed by 632 categories. The audio files are 10-second slices from Youtube videos. However, the annotations can be noisy since those are based on the user tags. FSD50K dataset includes a subset of 200 categories from Audioset ontology and has more than 50 thousand files. The SONYC-UST dataset also has weak labels of more than 50 hours of recordings from 60 different acoustic sensors.

Another option is to create synthetic audio mixtures using isolated sound events. This approach was adopted in the URBAN-SED dataset (Salamon et al., 2017), which contains synthesized soundscapes with sound event annotations generated using Scaper (Salamon et al., 2017) (a software library for soundscape synthesis). The original sound events are extracted from the UrbanSound8K dataset (Salamon et al., 2014), where a taxonomic categorization of urban sounds is proposed. At the top level, four groups are defined: HUMAN, NATURAL, MECHANICAL and MUSICAL, which have been used in previous works. The most frequent noise complaints in New York City from 2010 to 2014 were used to define the lower levels (Salamon et al., 2014).

Other datasets of urban sounds include only traffic noise (Abeßer et al., 2021), synthesized soundscapes from FSD50K (Abeßer, 2021), and acoustic scenes (Mesaros et al., 2018c) without information about sound sources. Datasets of environmental sound in other context have been also published for bioacoustic classifiers (Cramer et al., 2020), SED in domestic environments (Serizel et al., 2020), Audio captioning (Drossos et al., 2020) and machine monitoring (Tanabe et al., 2021). For a complete list of datasets, refer to `https://dcase-repo.github.io/dcase_datalist/`.

2.2.5 Evaluation metrics

Classification

Classification systems are typically evaluated using accuracy. Classification accuracy measures the number of correctly classified examples divided by the total amount of examples in the evaluation set (Mesaros et al., 2018b). This metric has been widely used in environmental sound classification (Salamon et al., 2014) and other related multi-class tasks such as acoustic scene classification (Mesaros et al., 2018b).

Sound event detection

The F-score ($F1$) and the Error Rate (ER) are the performance measures usually reported for SED systems (Salamon et al., 2017; Adavanne et al., 2016; Cakir et al., 2017), compared with ground-truth annotations on one–second length segments. For each segment, the False Positives (FP), False Negatives (FN) and True Positives (TP) rates are calculated; and then the precision and recall are computed as follows:

$$P = \frac{TP}{TP + FP}, \; R = \frac{TP}{TP + FN}. \tag{2.16}$$

$F1$ is the harmonic mean of P and R:

$$F1 = \frac{2PR}{P + R}. \tag{2.17}$$

The ER is calculated in terms of insertions (I), deletions (D) and substitutions (S). A substitution is defined as the case when the system detects an event on a segment, but with the wrong label. This is equivalent to have a FP and a FN in the same segment. After counting substitutions, the rest of FP are counted as insertions and the rest of FN as deletions. The ER measure is calculated as the integration of this values on the total number of segments K, as follows:

$$ER = \frac{\sum_{k=1}^{K} S(k) + \sum_{k=1}^{K} D(k) + \sum_{k=1}^{K} I(k)}{\sum_{k=1}^{K} N(k)}, \tag{2.18}$$

where $N(k)$ is the number of active classes in the ground–truth at the segment k (Mesaros et al., 2016b,a).

2.3 Interpretable deep learning

In this section, we present the literature review of methods for interpretable deep learning. We first define a taxonomy of methods based on previous works. We then explore the main proposed methods in each category in the taxonomy.

2.3.1 Taxonomy

Gilpin et al. (2018) propose a taxonomy of interpretable deep-learning methods. They define three different categories at the top level: explanations of the network processing, explaining network representations, and explanation-producing systems. The first two methods are post-hoc and aim to explain how the data is processed and represented in the neural network. Finally, the last type of method provides faithful explanations by devising models that are interpretable by design. Some other taxonomies propose to differentiate post-hoc and intrinsic methods (Barredo Arrieta et al., 2020). Besides, concept-based methods have gained much attention in the last few years, but those are not included explicitly in these taxonomies. These methods are devised to represent concepts in the latent space and can be both post-hoc and intrinsic. For example, post-hoc concept-based methods, such as Concept Activation Vectors (CAVs) (Kim et al., 2018), try to illustrate relevant concepts as vectors in the pre-trained latent space. On the other hand, intrinsic interpretable concept-based methods, such as Concept Whitening (Chen et al., 2020), are devised for aligning the latent space dimensions with pre-defined concepts.

Therefore we follow the taxonomy illustrated in Figure 2.7. Note that concept-based methods are in both parts of the diagram, post-hoc and intrinsic. Also, we included the case-based reasoning category within intrinsic methods since those methods are especially relevant in the context of this thesis, in particular prototype-based systems. Finally, note that we do not intend to present a complete taxonomy of interpretable deep-learning methods, but to combine the most used taxonomies in the research community to organize the literature review.

2.3.2 Post-hoc methods

Saliency maps

Saliency maps are the most used post-hoc methods for explaining the model's behavior for a particular instance. For example, the first proposed visualization methods (Simonyan et al., 2014) compute the gradients of the score predictions of the black-box CNN. Then, they use these gradients to find the portion inputs

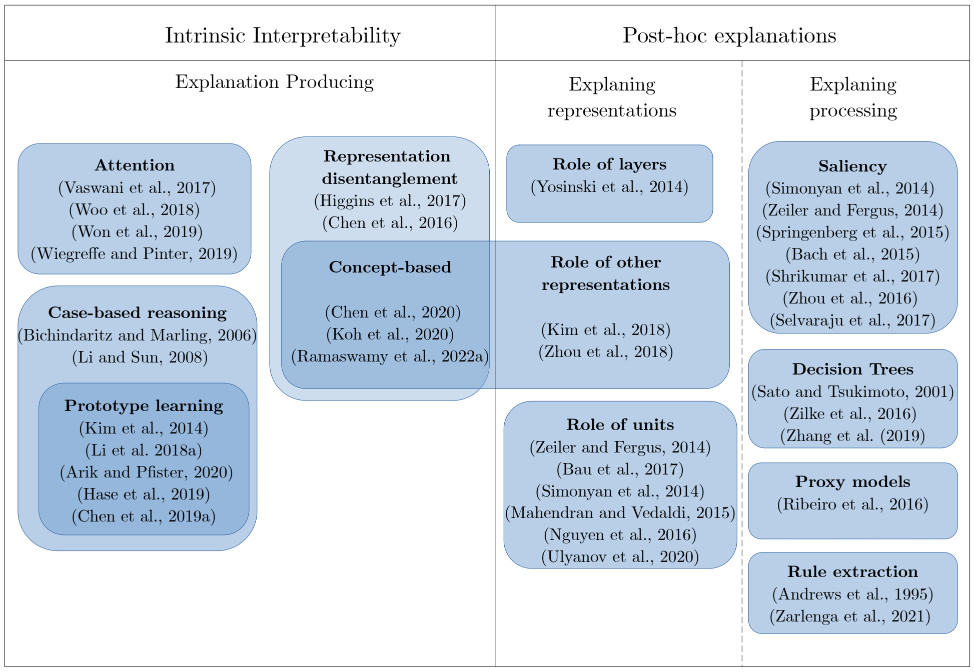


Figure 2.7: Alternative taxonomy of Interpretable Deep Learning. Extended from (Gilpin et al., 2018)

(pixels in images) that contribute most to the prediction. Unfortunately, the results of these methods are usually noisy, and therefore several works intend to improve the quality of the saliency maps by reducing the noise. For instance, DeConvNet (Zeiler & Fergus, 2014) and Guided Backprop (Springenberg et al., 2015) methods address this by guiding the gradients to emphasize the portions that positively improve the network's predictions.

Other methods integrate the gradients over some choices for the input image to better measure the pixel's contribution to the prediction. Given that gradients do not necessarily encode relevant information about the models' behavior, other approaches propagate other quantities through the network. For instance, Layer-wise Relevance Propagation (LRP) (Bach et al., 2015) and DeepLIFT (Shrikumar et al., 2017) methods propagate a relevance signal instead of a gradient. On the other hand, the Class Activation Maps (CAM) (Zhou et al., 2016) method linearly combines the activation of the last layer using the weights from the fully connected layer. Grad-CAM (Selvaraju et al., 2017) extends this method using the last layer's gradient instead of its weights.

Linear proxy models

Local interpretable model-agnostic explanation (LIME) learns a linear model to approximate the system's behavior for perturbations of a given input (Ribeiro et al., 2016). This linear model serves as a proxy for the black-box model in the space around the input. This method identifies the input regions that most influence the predictions across different models and domains.

Decision trees

Another proxy method to explain the model's decisions is disentangling neural representations into decision trees. CRED algorithm (Sato & Tsukimoto, 2001) transforms each output unit of a shallow neural network (with one hidden layer) into a decision tree. DeepRED (Zilke et al., 2016) extends the CRED algorithm to an arbitrary number of layers. In Zhang et al. (2019), decision trees quantitatively explain the logic for each prediction. These explanations are in the form of feature relevance.

Rule extraction

Like decision trees, automatic rule extraction is another well-studied approach to explaining neural networks (Gilpin et al., 2018). This type of method generates rules to characterize the model's behavior. The rules can take the form of simple if-then conditions or more complex combinations. These approaches provide both local (i.e., explaining a single prediction) and global explanations (i.e., explaining global patterns) (Barredo Arrieta et al., 2020). However, these methods, commonly called decomposition methods, have some limitations. For instance, most of them work only on shallow architectures (Andrews et al., 1995). The recently proposed ECLAIRE (Zarlenga et al., 2021) method tackles this issue by means of a two-step process that efficiently uses the latent space to guide its rule extraction.

Role of units and layers

Several post-hoc methods intend to explain the role of individual units or convolutional filters (Gilpin et al., 2018). They create a visualization of input patterns that maximize the response of a simple unit. Such works use datasets or generated instances to illustrate what activates a filter. For example, Zeiler & Fergus (2014) show the image patches from the dataset that most activate a given filter. Units can also be tested by their ability to solve a task. For example, the network dissection method (Bau et al., 2017) uses an annotated dataset of semantic concepts to quantify units' ability to segment these concepts.

Instead of using a datasets instance that activates a given filter or unit, (Simonyan et al., 2014) use optimization strategies to generate an input instance that highly activates the output of a given class. This technique is generally known as activation maximization. On the other hand, feature inversion methods (Mahendran & Vedaldi, 2015) generate input instances that match an intermediate activation hidden representation of the reference input. Besides, generative networks can be used to create images that explain the role of each layer (Nguyen et al., 2016; Ulyanov et al., 2020).

Moreover, Gilpin et al. (2018) argue that it is possible to obtain the role of a whole layer by testing its ability to solve different tasks from the one to which the network is an expert. This method is usually called transfer learning. Yosinski et al. (2014) propose a framework to quantify transfer learning capabilities. This quantitative approach estimates the feature generality or specificity.

Concept-based methods (role of other representations)

Instead of explaining the role of units or layers, concept-based methods intend to explain the role of another type of representation in the latent space. The idea is to represent other directions in the representation vector formed by the linear combination of individual units. For example, Concept Activation Vectors (CAVs) (Kim et al., 2018) are learnable vectors that represent high-level concepts in the latent space. A dataset with annotated concepts is used to learn linear classifiers in the latent representation. The vector representing a concept in the latent space is normal to the hyperplane separating examples with and without the concept. The interpretable Basis Decomposition (Zhou et al., 2018) method measures how different concepts contribute to a classification task.

However, Chen et al. (2020) show that these methods representing concepts as unit vectors may not work. They argue that to separate the concepts correctly, they should be near orthogonal in the latent space. They proposed an intrinsic interpretable method to solve this problem by making the latent space mean-centered and decorrelated.

2.3.3 Intrinsically interpretable methods

Concept-based methods

As previously mentioned, Chen et al. (2020) propose Concept Whitening (CW), a method to learn a mean-centered and decorrelated latent space to avoid the issues representing concepts as unit vectors. After centering and decorrelating the latent space, they apply a rotation to align the dimensions with the concepts. This method can be applied in every layer as a replacement for batch normalization. These transformations are learned during training using datasets of pre-defined concepts.

Other concept-based explanations methods use an intermediate representation to represent the concepts (Koh et al., 2020). These methods are known as concept bottlenecks. Given an input, the concept bottleneck model predicts a set of pre-defined concepts and then makes the predictions only based on that concepts. Ramaswamy et al. (2022) try this method in the context of image scene classification using semantic attributes present in the training dataset as concepts. They show that these attributes are not enough to explain the model behavior, so they propose to include a low-rank feature representation from the model's latent space.

Representation disentanglement

Concept-based methods disentangle the latent space by aligning hidden dimensions to pre-defined concepts. However, other methods for latent space disentanglement have been proposed (Gilpin et al., 2018). Latent space disentanglement is an old problem that has been tackled using techniques such as Principal Component Analysis (Jolliffe, 1986), Independent Component Analysis (Hyvärinen & Oja, 2000), and Nonnegative Matrix Factorization (Berry et al., 2007). However, deep neural networks can be tuned to learn disentangled representations. For instance, Beta-VAE (Higgins et al., 2017) is an autoencoder architecture that shows remarkable results. It is also possible to train generative adversarial models with an objective function to ensure disentanglement. For example, InfoGAN (Chen et al., 2016) maximizes the mutual information between the dimensions and the image observation. The experiments on MNIST dataset show that the latent space can encode the digit type, the rotation, and the width of the digits.

Attention mechanisms

Attention-based mechanisms are weighting functions that provide relative relevance to some parts of the input representation or features in hidden spaces (Gilpin et al., 2018). These mechanisms have proved very effective for natural language (Vaswani et al., 2017), computer vision (Woo et al., 2018), and audio-related (Won et al., 2019) problems. Although the output of an attention layer can be considered a form of explanation, it has been proved that these explanations are not necessarily reliable (Wiegreffe & Pinter, 2019). However, attention-based functions can be trained to explicitly extract the desired explanations (Gilpin et al., 2018; Ross et al., 2017). We follow this approach to use an attention mechanism in a sound event detection model to locate prototypes in the latent representations (see Chapter 6).

Case-based reasoning (prototype learning)

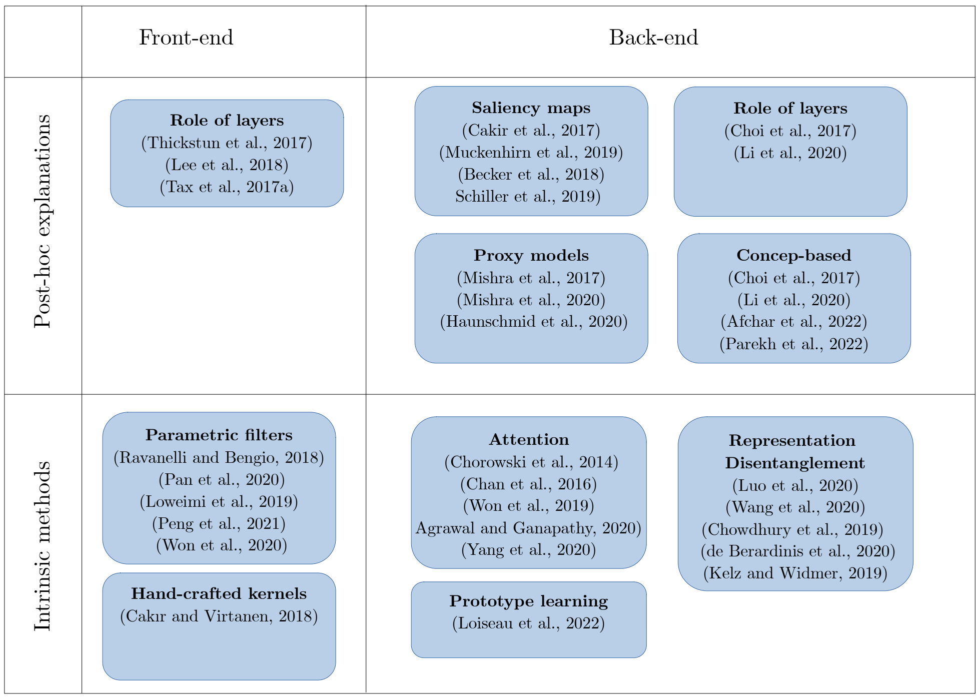
Case-based reasoning AI is based on the idea that a new experience can be model by representing, indexing, and organizing past cases (Aamodt & Plaza, 1994; Slade, 1991; Kim et al., 2014). This approach has been applied in the health sciences (Bichindaritz & Marling, 2006) and finances (Li & Sun, 2008). Prototype classification is a classical form of case-based reasoning, which is consistent with how humans explain their decision-making process, for instance, in complicated visual classification tasks (Kim et al., 2014).

Learning through prototypes is an approach that can provide inherent interpretability to deep neural networks. Decision are based on a few relevant examples known as *prototypes* that serve as a distillation of the data and have a high interpretable value (Bien & Tibshirani, 2011; Kim et al., 2014). A prototype is a vector that is close or identical to an instance of the training set. Deep neural networks can learn those prototypes in a flexible latent space. For example, the interpretable network proposed by Li et al. (2018a) for image classification is based on prototypes. The architecture appends a special prototype layer and uses an autoencoder. The prototypes are learned in the low-dimensional latent space produced by the encoder, and they can be reconstructed by applying the decoder. The predictions are based on the distance from the data instance to each prototype in the latent space. Thus, the explanations are the prototypes and the distances to them, which are the actual computations of the model to generate the output. This approach can be extended to hierarchical (Hase et al., 2019) and local prototypes that represent part of the images instead of the whole input (Chen et al., 2019a).

In Chapter 5 we extend this approach to audio classification. There, we propose the Audio Prototype Network (APNet) and show compelling results when applied to speech, music, and environmental audio, for problems with a single class label per audio clip. However, in a polyphonic setting (i.e multi-label SED), an input instance corresponding to several classes should be simultaneously close to prototypes of those classes in the latent space. Unfortunately, learning such latent space proved challenging in practice, thus motivating the alternative approach proposed in Chapter 6. This approach uses a combination of local prototypes and attention maps. Some other models combine attention mechanisms and prototypes. For instance, ProtoAttend (Arik & Pfister, 2020) selects input-dependent prototypes based on a relational attention mechanism that connects the encoded representation and the prototype candidates. In this case, the prototypes are instances from the training data. However, other methods use mean vectors as prototypes for few-shot learning (Sun et al., 2019).

2.4 Audio-based interpretable models

This section reviews previous research in explainable deep neural models for audio in relation to our work. Although the research in explainable machine learning is quickly expanding, the existing work that focuses on the audio domain is quite limited. Most of this research follows a post-hoc approach for explaining black-box models, and only a few works deal with intrinsically interpretable network architectures. Figure 2.8 shows a summary of the literature review. We separate the methods that aim to explain or interpret the first layers (front-end) from those that explain the back-end or the full model.



	Front-end	Back-end	
Post-hoc explanations	**Role of layers** (Thickstun et al., 2017) (Lee et al., 2018) (Tax et al., 2017a)	**Saliency maps** (Cakir et al., 2017) (Muckenhirn et al., 2019) (Becker et al., 2018) Schiller et al., 2019)	**Role of layers** (Choi et al., 2017) (Li et al., 2020)
		Proxy models (Mishra et al., 2017) (Mishra et al., 2020) (Haunschmid et al., 2020)	**Concep-based** (Choi et al., 2017) (Li et al., 2020) (Afchar et al., 2022) (Parekh et al., 2022)
Intrinsic methods	**Parametric filters** (Ravanelli and Bengio, 2018) (Pan et al., 2020) (Loweimi et al., 2019) (Peng et al., 2021) (Won et al., 2020)	**Attention** (Chorowski et al., 2014) (Chan et al., 2016) (Won et al., 2019) Agrawal and Ganapathy, 2020) (Yang et al., 2020)	**Representation Disentanglement** (Luo et al., 2020) (Wang et al., 2020) (Chowdhury et al., 2019) (de Berardinis et al., 2020) (Kelz and Widmer, 2019)
	Hand-crafted kernels (Cakır and Virtanen, 2018)	**Prototype learning** (Loiseau et al., 2022)	

Figure 2.8: Summary of interpretable models used in the audio domain.

2.4.1 Post-hoc methods

Regarding visualization methods for explainability, saliency maps were applied in (Cakir et al., 2017) to a convolutional-recurrent network trained for polyphonic sound event detection. The authors show that convolutional filters recognize specific patterns in the input and that the complexity of these patterns increases in the upper layers. A gradient-based approach was proposed in (Muckenhirn et al., 2019) for visualizing heat maps in raw waveform convolutional neural networks. Also, rules about relevance (Becker et al., 2018) and deep Taylor decomposition (Schiller et al., 2019) have been used to propagate the model's prediction backward. In contrast to the visualization methods, the explanations of the methods proposed in Chapters 5 and 6 come in the form of examples, which are easier to interpret by end-users of the model that may have no machine learning knowledge.

With regards to post-hoc methods that create proxy models, a variation of the LIME algorithm for audio content analysis—called SLIME—was proposed (Mishra et al., 2017; Mishra et al., 2020). It can generate explanations in the form of temporal, frequency, and time-frequency segmentation, and it was applied to singing voice detection. However, this kind of segmentations might not be suitable to represent harmonic or overlapping sounds; thus, Haunschmid et al. (2020) propose another adaptation of LIME, audioLIME, where the explanations are in the form of estimates from a source separation algorithm. In contrast, the models proposed in Chapters 5 and 6 are designed to be interpretable, and we do not generate explanations for black-box models. Furthermore, the time-frequency representations used as input for these models cannot be interpreted as simple images. We argue that example-based explanations are better suited for these audio representations than local pixels areas that do not always define sound objects.

The model internals, weights, neurons, and layers may also be explained. In the audio domain, such an approach using transfer learning was applied to study the capability of the layers of a pre-trained network to extract meaningful information for music classification and regression (Choi et al., 2017). The role of each layer in end-to-end speech recognition systems has been studied in (Li et al., 2020). The main idea is to synthesize speech signals from the hidden representations of each layer. The results show that specific characteristics of the speaker are gradually discarded in each layer, along with the ambient noise. Similar to these approaches, the network architectures proposed in this thesis are designed to generate prototypes by reconstructing representations that were learned in a latent space.

Concept-based methods have also been used in the audio domain. For instance, CAV vectors have been used to extract musical concepts describing the characteristics of a piece (Foscarin et al., 2022) or hierarchical concepts from playlists (Afchar et al., 2022). Moreover, Parekh et al. (2022) propose to use Non-negative Matrix Factorization (NMF) to obtain meaningful high-level listenable audio objects.

2.4.2 Intrinsic interpretable models

Regarding intrinsic interpretable network architectures, learning a disentangled latent space has been applied to learn interpretable latent factors that are related to pitch and timbre (Luo et al., 2020)—in the context of musical instrument recognition—and related to chord and texture (Wang et al., 2020)—in the context of generative models of polyphonic music. In addition, mid-level features (Chowdhury et al., 2019) and source separation (de Berardinis et al., 2020) have been used to improve the model interpretability for the problem of music emotion recognition. In (Kelz & Widmer, 2019), invertible networks (Ardizzone et al., 2019) have been applied for interpretable automatic polyphonic transcription. Similar to (Agrawal & Ganapathy, 2020), we use domain knowledge to design several parts of the network to learn interpretable representations for speech and audio signals.

The first models that used attention mechanisms in the audio domain applied them in conjunction with recurrent networks for speech recognition (Chorowski et al., 2014; Chan et al., 2016). Nowadays, attention mechanisms are widely used for speech, music, and other audio-related problems because of their ability to capture long-term temporal information. Self-attention mechanisms are used instead of recurrent layers to integrate temporal information; for instance, they were applied for music emotion recognition (Chaki et al., 2020; Jalal et al., 2020), music generation (Huang et al., 2019), and tagging (Won et al., 2019). Attention mechanisms can also be used for weighting the frequency dimension to create interpretable adaptive filter banks (Agrawal & Ganapathy, 2020). Recently, a visualization tool was proposed in (Yang et al., 2020) for understanding the attention mechanisms in self-supervised audio transformers. In contrast, our proposed model in Chapter 6 does not use attention maps to weight input's features. Instead, we use them as the only information to classify sound sources. Furthermore, since we devise the attention maps for proper reconstruction from the local prototypes, they are interpretable by design.

Prototypical learning has been applied to audio problems but not necessarily looking for interpretability. For example, prototypical networks (Snell et al., 2017) have been used for audio classification with little data (Pons et al., 2019) and bioacoustic few-show sound event detection (Nolasco et al., 2022). However, these systems are not intended to be interpretable, so one can not reconstruct the prototypes to the input space. In contrast, APNet, the model proposed in Chapter 5, and the model we propose in Chapter 6 allow for reconstructing the prototypes to the input space through the decoder and mapping them to the audio domain. A recent approach consists in learning the audio prototypes in the input space instead of the latent space and learn transformations (e.g., pitch shifting) to allow the signal reconstruction with few prototypes (Loiseau et al., 2022).

Interpretable front-end

In terms of interpretable network architectures, several parts of the network may be designed to target particular tasks, such as feature computation or signal decomposition. For instance, as shown in Section 1.5, the front-end of the network can be designed to extract interpretable representations. For example, it has been shown that the first layers of end-to-end convolutional neural networks that learn representations from raw audio data extract features that are similar to the spectrogram or energy in mel-frequency bands (Thickstun et al., 2017; Lee et al., 2018; Tax et al., 2017; Tokozume & Harada, 2017).

Additionally, some works have addressed the design of the first layers of these networks to tailor the feature extraction stage. For instance, parametric band-pass filters (sinc functions) have been used for speaker recognition (Ravanelli & Bengio, 2018) and for detecting neurodegenerative disorders (Pan et al., 2020). Moreover, Loweimi et al. (2019) extend this approach to other filters such as triangle filters (sinc^2 functions), gammatone, and gaussian kernels. Complex filters can be also used to obtain interpretable representations (Peng et al., 2021). Besides, parametric filters have been designed to extract harmonic-aware representations (Won et al., 2020).

On the other hand, trainable hand-crafted kernels can be designed to extract meaningful information (Cakır & Virtanen, 2018). In Chapter 4, we follow this approach to devise an end-to-end CNN for SED that extracts the energy of mel bands from the audio signal using a simple 1D convolutional layer. Furthermore, we show that these filters can be trained to obtain insights into the data.

CHAPTER 3

A library for reproducible sound recognition research

3.1 Introduction

As discussed in Section 1.6, most modern deep-learning models are considerably more complex than shallow models and heavily rely on the data and software used for their implementation (McFee et al., 2019). Therefore, it has become increasingly challenging to reproduce the findings or to compare a new method to earlier ones based only on the description of research systems found in publications (McFee et al., 2019; Six et al., 2018). For the optimal reuse of scientific research, numerous authors and institutions are advocating not only for open-access publications and data but also for the release of software and models (Colom et al., 2015; McFee et al., 2019; Bjornson, 2019).

Furthermore, both interpretable models and reproducible experiments are crucial to fostering a more responsible and trustworthy AI (High Level Expert Group on Artificial Intelligence, 2019; Barredo Arrieta et al., 2020). Therefore, a framework for the reproducible development and use of such systems is beneficial in the context of interpretable deep learning for audio-related problems.

In the light of the above, this chapter introduces `DCASE-models`, an open-source Python library developed in the context of this thesis work, whose main goal is to facilitate various aspects of the typical research pipeline of DCASE related problems with a particular emphasis on deep–learning models. The library has a careful design for easy extension and integration with other software tools. It offers an abstraction to common tasks, such as data preparation, data augmentation, feature extraction, model training, and evaluation. This allows for rapid prototyping of new methods and simplifies the efforts needed to release (and maintain) the code of a new model. Furthermore, the library aims to provide reference implementations of several baselines—including already trained models—to facilitate the comparison of methods. In this way, we strive for a low barrier to entry for students and researchers new to the field. Whenever possible, the library leverages from the original authors' implementations and existing software tools, such as `sed_eval` (Mesaros et al., 2016a) or VGGish (Hershey et al., 2017).

The package includes thorough documentation that covers the usage of the resources already available and describes the steps needed for extending the library, for instance, with new datasets, features and models. It also contains Python scripts, Jupyter Notebooks and a web interface, to illustrate the usefulness of the library.

3.2 Design principles and practices

The library provides a simple and lightweight set of basic components that are generally part of a computational environmental audio analysis system. Users can exploit the library functions to tackle various tasks—such as acoustic scene classification, sound event detection, and audio tagging—while experimenting with improvements or extensions of its components.

Apart from a collection of functions for dataset handling, data preparation, feature extraction, and evaluation (most of which rely on existing tools), `DCASE-models` includes a model interface to standardize the interaction of machine learning methods with the other system components. This also provides an abstraction layer to make the library independent of the backend used to implement the machine learning model (e.g. Keras,[3] PyTorch,[4] TensorFlow,[5] Scikit-learn[6]). The standardized behavior of the machine learning method implementation allows the comparison of different models in a straightforward manner. The library currently includes Keras implementations of several deep–learning models reported in the literature, which are ready to use with minimal effort. As a result, one can get an application with a few lines of code (see Section 3.4).

Regarding the usability of the library, the design and implementation are aimed to make it easy to learn and use. It is organized in a flat package layout with classes that define concise interfaces. All functions are thoroughly documented and include example code that demonstrates their usage. Besides, we follow PEP-8 recommendations to make sure code is readable and easy to follow. The documentation of the library is prepared using Sphinx and includes clear instructions on how to extend different components. The latest stable release can be smoothly installed from PyPI and only has requirements of other well-known, portable and tested packages.

[3]`https://keras.io`
[4]`https://pytorch.org`
[5]`https://www.tensorflow.org`
[6]`https://scikit-learn.org`

Considering the sustainability and maintainability of the library, we strive for adopting modern open–source software development practices, as suggested in McFee et al. (2019). The code is released under the MIT license and all development is conducted on GitHub. This makes the project readily accessible for the community to use, test, contribute, and bring support. It also helps the release of updates by keeping a record of the changes and versions and incorporate other software development services, such as continuous integration testing.

3.3 Library organization and description

Figure 3.1 shows a diagram of `DCASE-models` main classes, which also includes some specializations of each base class that are available. Next, a description of the main classes and functionalities is presented, following the order of the typical pipeline: dataset preparation (3.3.1); data augmentation (3.3.2); feature extraction (3.3.3); data loading (3.3.4); data scaling (3.3.5); and model handling (3.3.6).

3.3.1 Dataset

This is the base class designed to manage a dataset, its paths, and its internal structure. It includes methods to download the data, resample the audio files, and check that both processes succeed.

The library covers several publicly available datasets related to different tasks. At the moment, these are: ESC (Piczak, 2015) and UrbanSound8k (Salamon et al., 2014) for audio classification; TAU Urban Acoustic Scenes 2019 and 2020 (Mesaros et al., 2018c; Heittola et al., 2020) for acoustic scene classification; URBAN-SED (Salamon et al., 2017), TUT Sound Events 2017 (Mesaros et al., 2017) and MAVD-traffic (Zinemanas et al., 2019b) for sound event detection; and FSDKaggle2018 (Fonseca et al., 2018) and SONYC-UST (Cartwright et al., 2019) for audio tagging. In next releases of the library other relevant datasets will be included.

Each dataset is implemented in the library as a class that inherits from `Dataset`. This design provides a common and simple interface to work with any dataset. For instance, to use the UrbanSound8k dataset, it is enough to initialize its class with the path to the data folder, as follows.

```
>>> dataset = UrbanSound8k(DATASET_PATH)
```

Then, the following methods are used to download the dataset and change its sampling rate (to 22050 Hz).

```
>>> dataset.download()
>>> dataset.change_sampling_rate(22050)
```

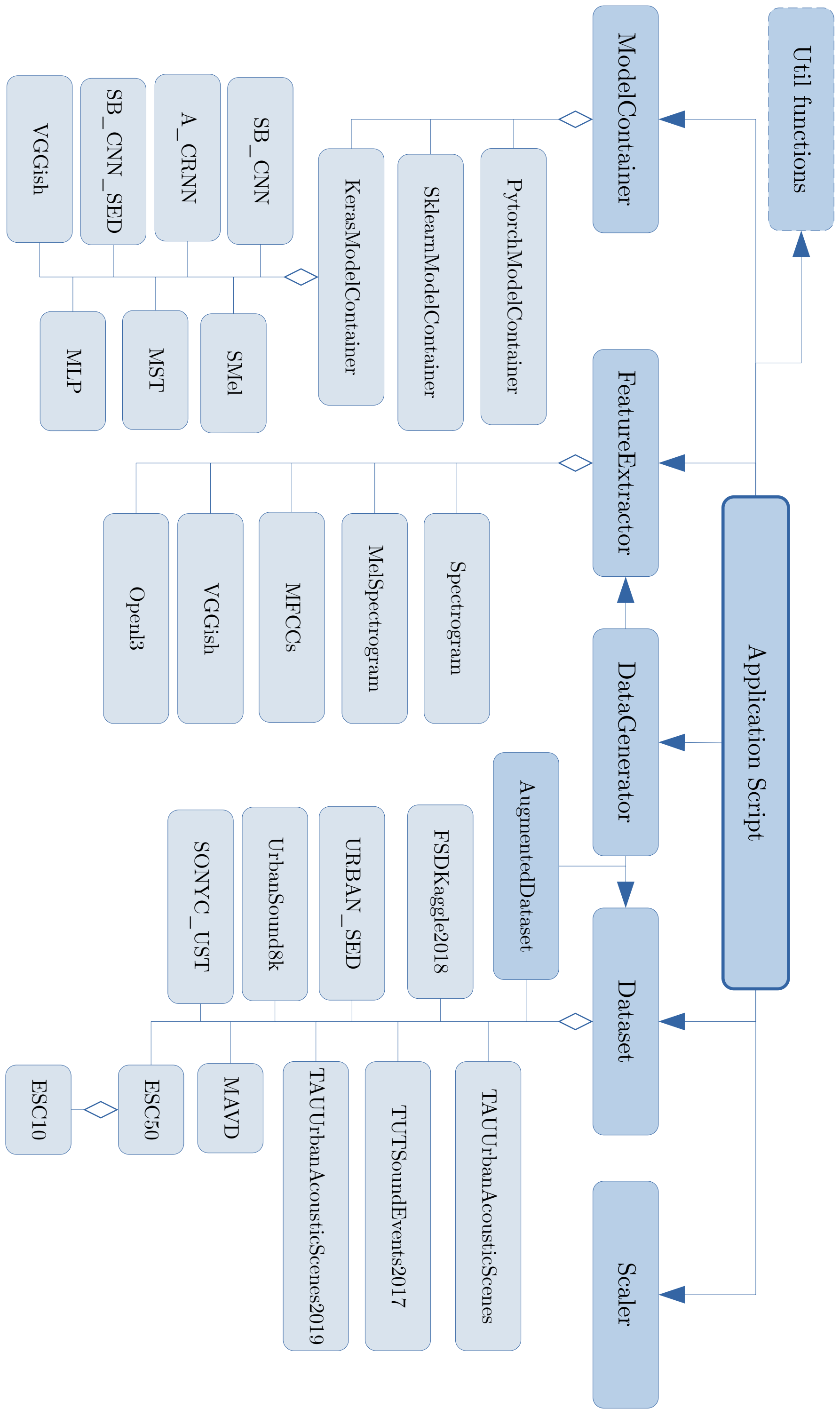


Figure 3.1: Class diagram of DCASE-models showing all base classes and some of the implemented specializations.

Most of the datasets devised for research include a fold split and a corresponding evaluation setup (e.g. 5–fold cross–validation). This fold split is generally carefully selected to avoid biases and data contamination (Witten & Frank, 2005). In order to keep the results comparable to those reported in the literature, `DCASE-models` uses, whenever available, the predefined splits for each dataset. However, the user may define different splits or evaluation setups if needed.

3.3.2 AugmentedDataset

The previously defined `dataset` instance can be expanded using data augmentation techniques. The augmentations implemented so far are pitch–shifting, time–stretching, and white noise addition. The first two are carried out by means of `pysox` (Bittner et al., 2016).

An augmented version of a given dataset can be obtained by initializing an instance of the `AugmentedDataset` class with the `dataset` as a parameter, as well as a dictionary containing the name and parameters of each transformation.

```
>>> augmentations = [
        {'type': 'pitch_shift', 'n_semitones': -1},
        {'type': 'time_stretching', 'factor': 1.05},
        {'type': 'white_noise', 'snr': 60}
    ]
>>> aug_dataset = AugmentedDataset(
        dataset,
        augmentations
    )
```

After initialization, the following method will perform the actual augmentation and create new audio files for every dataset element according to the type and parameters of each augmentation.

```
>>> aug_dataset.process()
```

The augmented dataset is indeed an instance of `Dataset`, so it can be used as any other dataset in the following steps of the pipeline.

3.3.3 FeatureExtractor

This is the base class to define different types of feature representations. It has methods to load an audio file, extract features, and save them. It can also check if the features were already extracted.

Four types of feature representations have been implemented as specializations of the base class, namely Spectrogram, MelSpectrogram, MFCCs, VGGish (Hershey et al., 2017), and Openl3 (Cramer et al., 2019). The first three are classic time–frequency representations (see Section 2.2.1) which are implemented using `librosa` functions. The last two are pre–trained neural–network–based models that extract embeddings from the audio signal. Openl3 is a recently proposed neural network trained on audio–visual information. VGGish is a deep convolutional neural network trained on the Audioset dataset that is also used as a feature extractor.

A `FeatureExtractor` is initialized with some parameters. For instance, to define a `Spectrogram` feature extractor the parameters are: length and hop in seconds of the feature representation analysis windows (model's input); window length and hop size (in samples) for the Short-time Fourier Transform (STFT) calculation; and the sampling rate. If the audio files are not sampled at this frequency, they are converted before calculating the features.

```
>>> features = Spectrogram(
        sequence_time=1.0,
        sequence_hop_time=0.5,
        audio_win=1024,
        audio_hop=512,
        sr=22050
    )
```

After initialization, the following method computes the features for each audio file in the dataset.

```
>>> features.extract(dataset)
```

Once the features are extracted and saved to disk, they can be loaded using `DataGenerator` as explained in the following.

3.3.4 DataGenerator

This class uses instances of `Dataset` and `FeatureExtractor` to prepare the data for model training, validation and testing. An instance of this class is created for each of these processes.

```
>>> data_gen_train = DataGenerator(
        dataset,
        features,
        train=True,
        folds=['train']
    )
```

```
>>> data_gen_val = DataGenerator(
        dataset,
        features,
        train=False,
        folds=['val']
    )
```

At this point of the pipeline, the features and the annotations for training the model can be obtained as follows.

```
>>> X_train, Y_train = data_gen_train.get_data()
```

Additionally, instances of `DataGenerator` can be used to load data in batches. This feature is especially useful for training models on systems with memory limitations.

3.3.5 Scaler

Before feeding data to a model, it is common to normalize the data or scale it to a fixed minimum and maximum value. To do this, the library contains a `Scaler` class, based on `scikit-learn` preprocessing functions, that includes `fit` and `transform` methods.

```
>>> scaler = Scaler("standard")
>>> scaler.fit(X_train)
>>> X_train = scaler.transform(X_train)
```

In addition, the scaler can be fitted in batches by means of passing the `Data-Generator` instance instead of the data itself.

```
>>> scaler.fit(data_gen_train)
```

It is also possible to scale the data as it is being loaded from the disk, for instance, when training the model. To do so, the `Scaler` can be passed to the `DataGenerator` after its initialization.

```
>>> data_gen_val.set_scaler(scaler)
```

3.3.6 ModelContainer

This class defines an interface to standardize the behavior of machine learning models. It stores the architecture and the parameters of the model. It provides methods to train and evaluate the model, and to save and load its architecture and weights. It also allows the inspection of the output of its intermediate stages (i.e. layers).

The library also provides a container class to define Keras models, namely `KerasModelContainer`, that inherits from `ModelContainer`, and implements its functionality using this specific machine learning backend. Even though the library currently supports only Keras, it is easy to specialize the `ModelContainer` class to integrate other machine learning tools, such as PyTorch.

Each model has its own class that inherits from a specific `ModelContainer`, such as `KerasModelContainer`. The models currently implemented using Keras are: Multi–layer Perceptron (MLP), SB-CNN (Salamon & Bello, 2017), SB-CNN-SED (Salamon et al., 2017), A-CRNN (Adavanne et al., 2017), MST (Tax et al., 2017), SMel (Zinemanas et al., 2019a) and VGGish (Hershey et al., 2017). Other models are in preparation for inclusion in next releases of the library.

A model's container has to be initialized with some parameters.

```
>>> model_cont = SB_CNN(**model_params)
```

These parameters vary across models, among which the most important are: input shape, number of classes, and evaluation metrics. Specific parameters may include the number of hidden layers or the number of convolutional layers, among others. The `ModelContainer` class has a method to train the model.

```
>>> model_cont.train(
        (X_train, Y_train),
        **train_params
    )
```

Training parameters can include, for example, number of epochs, learning rate and batch size. To train the model in batches, the `DataGenerator` object can be passed to the `train` method instead of the pre–loaded data.

```
>>> model_cont.train(data_gen_train, **train_params)
```

Performing model evaluation is also simple. For instance, the following code uses the test set for evaluating the model.

```
>>> data_gen_test = DataGenerator(
        dataset,
        features,
        train=False,
        folds=['test']
    )
>>> X_test, Y_test = data_gen_test.get_data()
>>> results = model_cont.evaluate((X_test, Y_test))
```

The results' format depends on which metrics are used. By default, the evaluation is performed using metrics available from the `sed_eval` library (Mesaros et al., 2016a). Therefore, the results are presented accordingly. Nevertheless, `DCASE-models` enables the use of others evaluating frameworks such as `psds_eval` (Bilen et al., 2020), or the use of user–defined metrics in a straightforward way.

When building deep–learning models it is common practice to use fine–tuning and transfer learning techniques. In this way, one can reuse a network that was previously trained on another dataset or for another task, and adapt it to the problem at hand. This type of approach can also be carried out with the `ModelContainer`.

3.4 Application examples

The Python package of the library includes a set of examples, organized into three different categories, which illustrate the usefulness of `DCASE-models` for carrying out research experiments or developing applications. These examples can also be used as templates to be adapted for implementing specific DCASE methods.

Firstly, some Python scripts are provided, that perform each step in the typical development pipeline of a DCASE task, i.e downloading a dataset, data augmentation, feature extraction, model training, fine–tuning, and model evaluation. See the documentation of the library for a tutorial that follows all these examples.

Secondly, several Jupyter Notebooks are also included whose aim is to replicate some of the experiments reported in the literature using `DCASE-models`, in particular those in (Adavanne et al., 2017; Salamon & Bello, 2017; Salamon et al., 2017; Zinemanas et al., 2019b,a). For instances, the following script shows that the experiments from (Salamon & Bello, 2017) can be performed in few lines of code:

```
>>> from dcase_models.data.data_generator import DataGenerator
>>> from dcase_models.data.datasets import UrbanSound8k
>>> from dcase_models.data.scaler import Scaler
>>> from dcase_models.data.data_augmentation import AugmentedDataset
>>> from dcase_models.data.features import MelSpectrogram
>>> from dcase_models.model.models import SB_CNN

>>> dataset = UrbanSound8k(DATASET_PATH)
>>> dataset.download()

>>> semitones = [-2, -1, 1, 2]
>>> params_augmentation  = [
        {'type' : 'pitch_shift', 'n_semitones': s } for s in semitones
    ]
>>> sr = 44100

>>> aug_dataset = AugmentedDataset(
        dataset,
        sr,
        params_augmentation
    )
>>> aug_dataset.process()

>>> params_features = {
        "mel_bands": 128,
        "n_fft": 1024
        "audio_hop": 1024,
        "audio_win": 1024,
        "sequence_hop_time": 1.0,
        "sequence_time": 3.0,
        "sr": sr
    }

>>> features = MelSpectrogram(**params_features)

>>> if not features.check_if_extracted(aug_dataset):
        features.extract(aug_dataset)

>>> folds_train, folds_val, folds_test = evaluation_setup(
        'fold1',
        dataset.fold_list,
        "cross-validation"
    )
>>> data_gen_train = DataGenerator(
        aug_dataset,
        features,
        folds=folds_train,
        train=True,
        scaler=None
    )

>>> scaler = Scaler(normalizer="standard")
>>> scaler.fit(data_gen_train)
>>> data_gen_train.set_scaler(scaler)
>>> data_gen_val = DataGenerator(
        aug_dataset,
        features,
        folds=folds_val,
        train=False,
        scaler=scaler
    )

>>> features_shape = features.get_shape()
>>> n_frames_cnn = features_shape[1]
>>> n_freq_cnn = features_shape[2]
>>> n_classes = len(dataset.label_list)

>>> model_container = SB_CNN(
        n_classes=n_classes,
        n_frames_cnn=n_frames_cnn,
        n_freq_cnn=n_freq_cnn
    )

>>> params_train = {
        batch_size=128,
        epochs=100
    }
>>> data_train = data_gen_train.get_data()
>>> data_val = data_gen_val.get_data()
>>> model_container.train(
        data_train,
        data_val,
        label_list=dataset.label_list,
        weights_path=exp_folder,
        **params_train
    )
```

Finally, a web interface for sound classification is also included as a proof of concept of the potential of `DCASE-models` to build high–level applications for computational environmental audio analysis. It gives access to most of the library's functionalities through a graphical user interface. Besides, it provides visualization tools to explore the dataset and to inspect the errors made by the model. It is also possible to listen to the audio files of the dataset and to test the model on an audio file provided by the user. Figure 3.2 shows a screenshot of the web application. The HTML front–end is developed with the `dash` library: `https://plotly.com/dash/`.

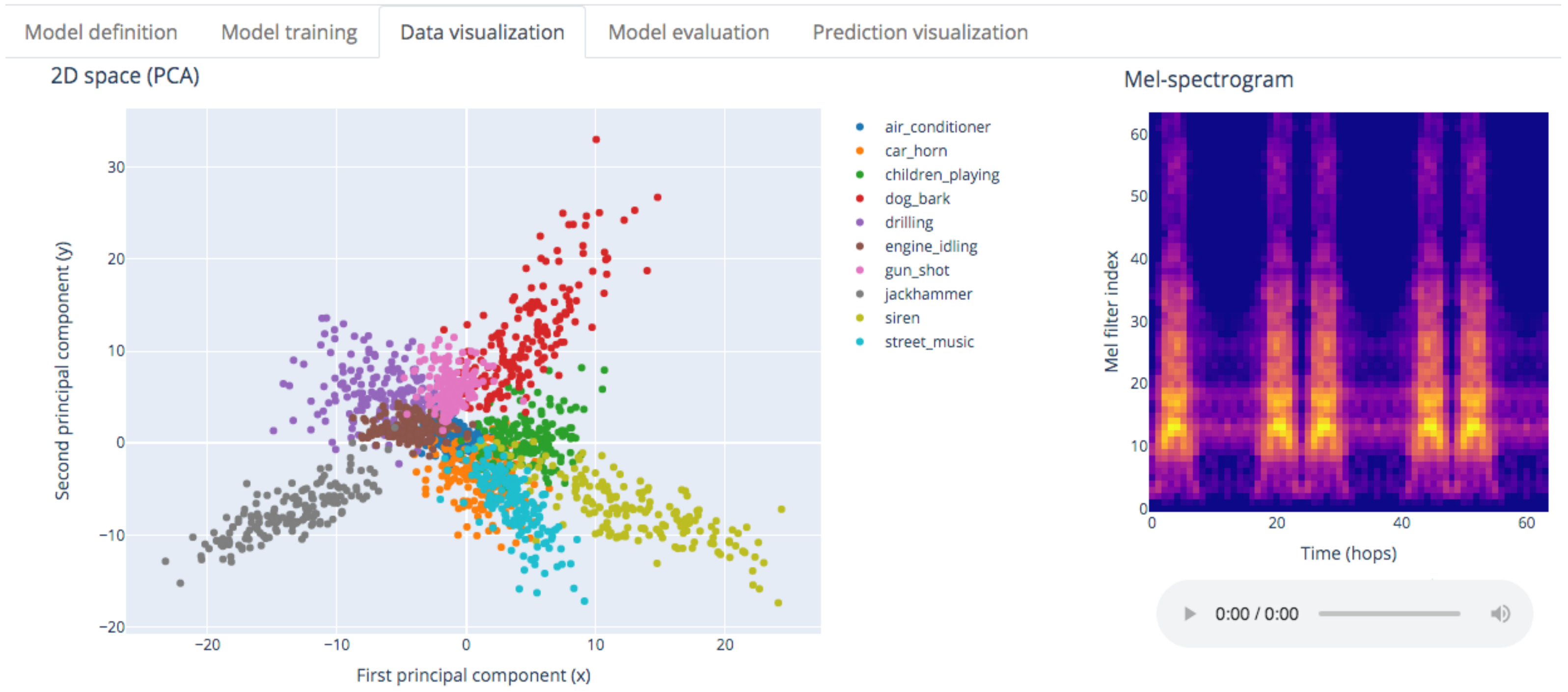


Figure 3.2: Screenshot of the web application for sound classification developed with `DCASE-models` as backend. In the *Data visualization* tab, the user can explore the training set by visualizing a 2D projection (principal component analysis, PCA) space of some model's intermediate output. It is also possible to inspect wrongly classified instances, visualize the feature representation, and listen to the audio files.

3.5 Discussion

This chapter describes `DCASE-models`[7], a Python library developed during this thesis work. This open–source library provides a number of classes useful for rapid prototyping solutions for DCASE related problems, with a particular emphasis on deep–learning models. The library has a flat and light design that allows easy extension and integration with other existing tools. The design also provides an abstraction layer that seeks to mitigate the impact of changes in the backends used for implementing the machine learning models. We put considerable effort into the usability aspects of the library to encourage its adoption by the DCASE community. In particular, we hope it turns out helpful for those students and researchers new to the field. In this sense, we look forward to other researchers' feedback and contributions. Additionally, we believe that the library could simplify the process of releasing and maintaining the code of new models. This, in turn, could improve research reproducibility and simplify methods comparison. To this respect, the package includes a set of application examples, some of which replicate a number of experiments reported in the literature. Besides, a web interface for sound classification is provided as a proof of concept of the usefulness of the library for developing high–level applications for computational analysis of acoustic scenes and sound events.

In addition to the applications presented in this chapter, the DCASE-models library has been a very useful software tool for this thesis. For instance, it has been used for developing and training the models of Chapters 5 and 6. The experiments of Chapter 4 were performed using ad-hoc implementations, but the models are available in the DCASE-models library. Besides, the web interface for sound classification included in the library, was the base for the visualization tools proposed in Chapter 5. Moreover, this library has eased the development of refining and debugging methods for the model presented in Chapter 5.

[7]`https://github.com/MTG/DCASE-models`

CHAPTER 4

An interpretable front-end

4.1 Introduction

As mentioned in Chapter 1, end-to-end models can be trained to obtain the predictions directly from the waveform signal (Section 1.5). These models have been applied to speech recognition (Tüske et al., 2014; Zeghidour et al., 2018; Sainath et al., 2017), speaker recognition (Valenti et al., 2018), automatic music labeling (Lee et al., 2018; Dieleman & Schrauwen, 2014), music audio tagging (Pons et al., 2018), and automatic notes transcription (Thickstun et al., 2017). But, even though end-to-end image processing has brought excellent results to the image classification task (e.g. AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2015), and GoogLeNet (Szegedy et al., 2015)), the results yielded by end-to-end audio processing are not better than those of the models whose input is a time-frequency representation (Tax et al., 2017).

In end-to-end neural networks, feature extraction is usually done by the first convolutional layers. As mentioned in Chapter 1, generally, these models are black boxes, intending to make the network learn the acoustic features that better discriminate the classes. However, it is possible to use domain knowledge to tailor the feature-extraction layers of the network to a particular problem. For instance, parametric band-pass filters (sinc functions) have been used for speaker recognition (Ravanelli & Bengio, 2018) and for detecting neurodegenerative disorders (Pan et al., 2020). Moreover, Loweimi et al. (2019) extend this approach to other filters such as triangle filters (sinc^2 functions), gammatone, and gaussian kernels. Complex filters can also be used to obtain interpretable representations (Peng et al., 2021). Besides, parametric filters have been designed to extract harmonic-aware representations (Won et al., 2020).

The mel-spectrogram transformation (MST) model is another example of this approach (Tax et al., 2017), in which the input is one second of the audio signal and the target output is the log-mel-spectrogram. If this model is concatenated with a neural network whose input is a time-frequency representation, it forms an end-to-end neural network. The first layers of the network can still be trained to adapt the feature extraction to a particular problem, but starting at an initial condition that has proved to be effective for the problem domain. As a result, the training may also need a smaller amount of data and fewer training epochs. On top of this, the first layers of the network (either with the initial values or after training) could be applied to other similar tasks in audio processing.

This chapter aims to apply the end-to-end approach to the SED problem by incorporating interpretable layers devised using domain knowledge. To achieve this, we propose a novel scheme, SMel, to compute the energy of the mel-spectrogram using a neural network architecture. We show that the proposed SMel scheme yields better results than the MST model (Tax et al., 2017). Then, we concatenate the SMel model with a state-of-the-art CNN for urban sound event detection (Salamon et al., 2017) to form the end-to-end architecture. A similar approach based on end-to-end neural networks to tackle the SED problem was proposed in (Cakır & Virtanen, 2018). It uses a learned time-frequency representation as the input of a convolutional-recurrent network. However, in that work, the feature extraction network is implemented by calculating the real and the imaginary parts of the discrete Fourier transform. Besides, the focus of that work is not urban environments.

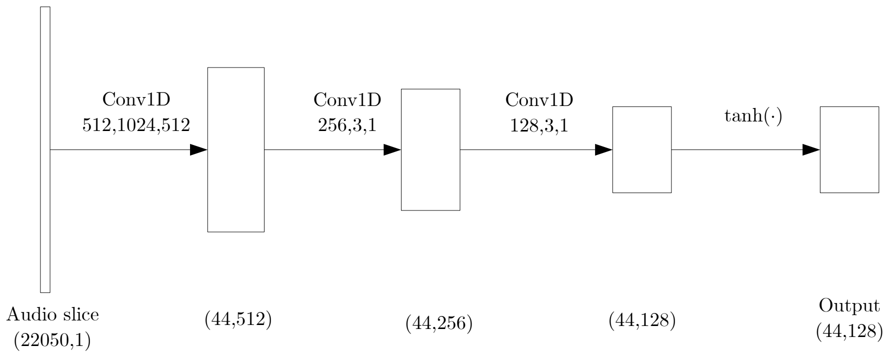
In this chapter, we also implement PCEN (see Section 2.2.1) as a neural network layer, and we study its applicability to the SED task. We show how the network can learn the PCEN parameter values and how the mel-frequency filter bank changes after training.

4.2 Calculating the log-mel-spectrogram

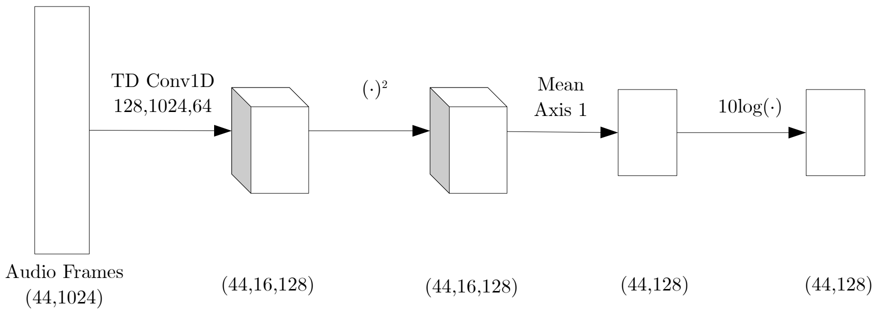
This section presents a novel model to calculate the log-scaled mel-spectrogram based on a simple mel filter bank. First, we present a recently published work on this topic and then explain our proposed model. Finally, we compare both architectures. The input of both models are one-second length non-overlapping slices of the audio signal, and the output is the log-scaled mel-spectrogram.

4.2.1 MST model

MST model is a CNN architecture devised to calculate the log-scaled mel-spectrogram of an audio slice (Tax et al., 2017). The architecture is formed by three convolutional layers of 512, 256 and N_{mels} filters respectively, where N_{mels} is the number of mel bands. Fig. 4.1a shows the diagram of the MST model for $N_{mels} = 128$, window length of 1024 points, a hop of 512 points, and sampling rate $f_s = 22050$ Hz.



(a) MST model.



(b) SMel model.

Figure 4.1: Block diagram of (a) MST model and (b) the proposed model for $N_{mels} = 128$, $f_s = 22050$, window length of 1024 points and hop of 512 points. In convolution layers the parameters showed are the number of filters, the size of the kernels and the stride value in that order. Above the arrows are shown the signals' dimensions.

4.2.2 Proposed model (SMel)

We propose a simpler approach, namely the SMel model, which is based on the fact that all the steps to calculate the mel-band energy are differentiable functions. Therefore, those steps can be implemented as layers of a neural network. The input of our network is a matrix whose columns are the frames of the audio signal multiplied by a Hann window. The first layer of the network is a time-distributed (TD) convolution of N_{mels} filters and it is initialized with a mel filter bank. Therefore, the output of the first layer is the result of the filter bank applied to each signal frame. In the next layers, the energy of each band is calculated by an element-wise square function; a mean value function; and a logarithmic function to convert energy values to decibels (see Fig. 4.1b).

The mel filter bank is formed by N_{mels} filters with triangular frequency response centered in the mel-scale frequencies and overlapped by half of their bandwidth. Therefore, the frequency response of filter l is:

$$H_l(f) = \Lambda \left(\frac{f - f_l}{\Delta f_l} \right) \text{ for } f \geq 0, \tag{4.1}$$

where f_l is the central frequency and $\Delta f_l = f_{l+1} - f_l$ is half bandwidth. Therefore, we design impulse responses for each filter as follows:

$$h_l(t) = 2\Delta f_l \text{sinc}^2(t\Delta f_l) \cos(2\pi f_l t) w(t), \tag{4.2}$$

where $w(t)$ is a Hann window.

4.2.3 Comparing models

To compare both models, we train them with the same dataset and parameters. The dataset used is the URBAN-SED, that contains audio files with urban sound events. This dataset includes 6000 files of 10 seconds for training, and 2000 files for validation and test (Salamon et al., 2017). It is devised for polyphonic urban SED and includes ten classes from mechanical (e.g. air conditioner, engine idling), human (e.g. children playing), musical (e.g. street music), and natural (e.g. dog bark) categories.

We down-sample the audio files to a sampling rate of 22050 Hz, just to decrease the computational cost, assuming this provides a sufficient bandwidth for the problem at hand. We process the audio signal in short-time windows of length $N = 1024$ samples and using a hop size of 512 samples. The target function (ground truth of the mel-spectrogram) is calculated using *librosa* (McFee et al., 2015) with 128 mel bands from 0 Hz to 11025 Hz.

To train the MST model we use the same strategy proposed by its authors (Tax et al., 2017). To train our model it is especially important to carefully choose the learning rate because the logarithmic function has a large gradient close to zero. We use the gradient descent equation to estimate the learning rate for the worst case. As the *librosa* function used to convert power to decibels saturates on -100 dB (power equal to 10^{-10}), that is our worst case. So, we estimate the learning rate to have a small relative change on $x = 10^{-10}$.

We train both models for 100 epochs using *Adam* optimizer and a mean squared loss function. Although theoretically, with the initialized parameters, SMel extracts the mel-spectrogram almost exactly, it is interesting to see the variation of the loss function for each model, as shown in Fig. 4.2. It is clear that the approximation of our model is better than MST. Furthermore, due to the initialization of the filters, the convergence of the proposed model is faster. Fig. 4.3 shows the output of each model for a randomly selected file of the validation set. Note that the output of MST model is more blurry, particularly at high frequencies.

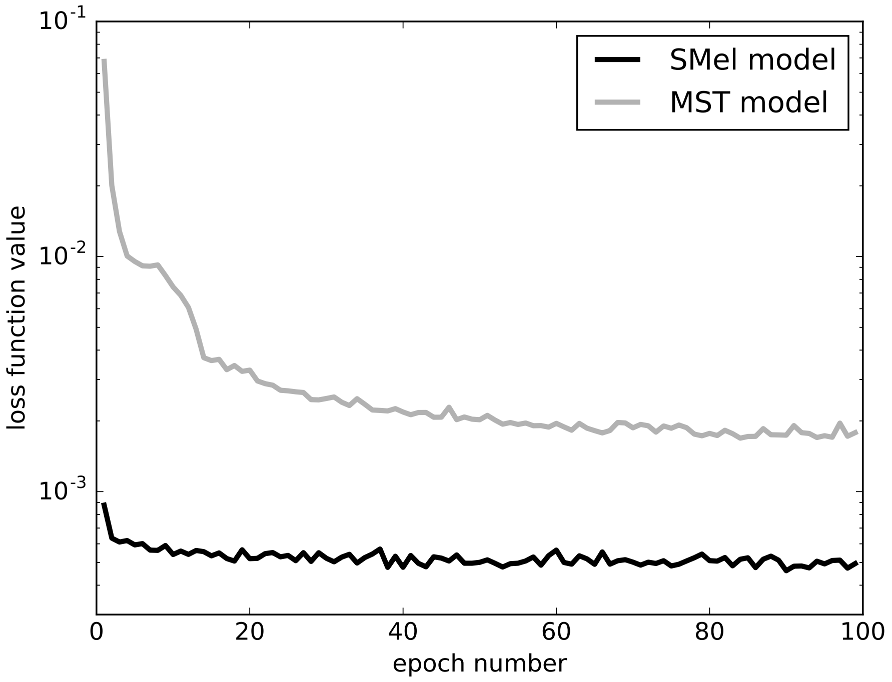


Figure 4.2: Variation of the loss function value for SMel and MST in training.

4.3 Energy Normalization

We implement PCEN (see Eq. 2.6) with neural network layers including frequency-dependent parameters:

$$E_{PCEN}\left[i,l\right] = \left(\frac{E\left[i,l\right]}{\left(\epsilon[l] + M\left[i,l\right]\right)^{\alpha[l]}} + \delta[l] \right)^{r[l]} - \delta[l]^{r[l]}, \tag{4.3}$$

where $M\left[i,l\right] = \left(E^t * \phi_T\right)\left[i,l\right]$, while $\alpha[l]$, $\epsilon[l]$, $r[l]$ and $\delta[l]$ are the frequency-dependent positive constants. The low-pass filter is implemented as a first order IIR filter as follows:

$$M\left[i,l\right] = (1 - s[l])M\left[i-1,l\right] + s[l]E\left[i,l\right], \tag{4.4}$$

where $s[l]$ is a frequency-dependent smoothing coefficient (Wang et al., 2017b). $M\left[i,l\right]$ is calculated using a recurrent layer that implements the IIR filter that has been proposed in Wang et al. (2017b). The rest of the operations are implemented in a layer that has two inputs $E\left[i,l\right]$ and $M\left[i,l\right]$. Fig. 4.4 shows the diagram of this implementation.

In the next section, we show the benefits of using the SMel model and the PCEN normalization for sound event detection in urban environments.

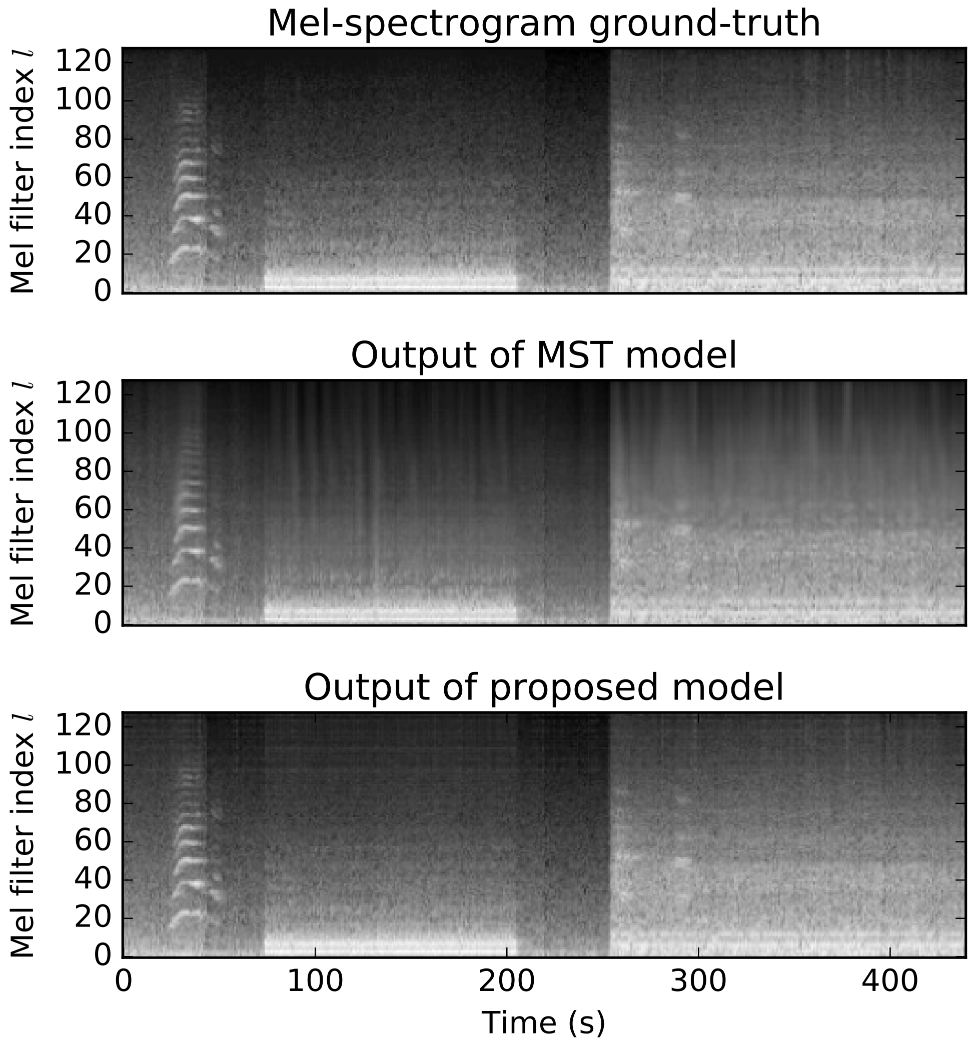


Figure 4.3: The mel-spectrogram ground-truth calculated using *librosa* implementation, and the outputs of the MST and SMel models for an example file from the validation set.

4.4 Experiments and Results

In order to show how it is possible to use the proposed model, we concatenate it to a network that uses the mel-spectrogram as input. We use the CNN described on URBAN-SED article (Salamon et al., 2017) as the baseline. This network has three 2D convolutional layers followed by three fully-connected layers. The final layer is a sigmoid of 10 units that perform the classification task.

We train three networks: (CNN) the baseline (mel-spectrogram from *librosa* as input) (Salamon et al., 2017); (MST+CNN) the MST model concatenated

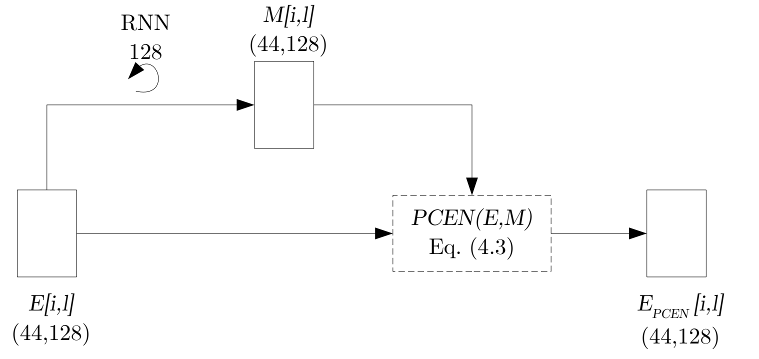


Figure 4.4: Diagram of PCEN implementation with neural network.

to the baseline CNN; and (SMel+CNN) the proposed model also concatenated to the CNN. All models are implemented using *keras* (Chollet et al., 2015) library with Tensor Flow as backend. This scheme implies that the temporal resolution of the detected sound events is one second.

4.4.1 Training strategy

In order to train the CNN, we use the same strategy as presented in Salamon et al. (2017) and the parameter values presented in Section 4.3. To train the MST+CNN and SMel+CNN networks we use a strategy inspired by Branch Training for hierarchical classification (Zhu & Bain, 2017). The training process is performed with two loss functions; mean squared for mel-spectrogram (l_0) and binary cross entropy for classification (l_1). The final loss function (l) is a weighted sum of the two losses:

$$l = w_0 l_0 + w_1 l_1, \tag{4.5}$$

where $[w_0, w_1]$ is a pair of weights. We set w_0 with a small value, as a way to regularize the mel-spectrogram training, and $w_1 = 1 - w_0$.

4.4.2 Classification results

We train the three networks for 100 epochs using the strategies proposed previously and using *Adam* optimizer. Fig. 4.5 shows the variations of $F1$ value on the validation set for the three networks. We save the network's weights on the epoch for which the F1 value on the validation set is maximum. Fig. 4.6 shows the best attained F1 values per class on the test set for each model. Table 4.1 shows the overall detection results.

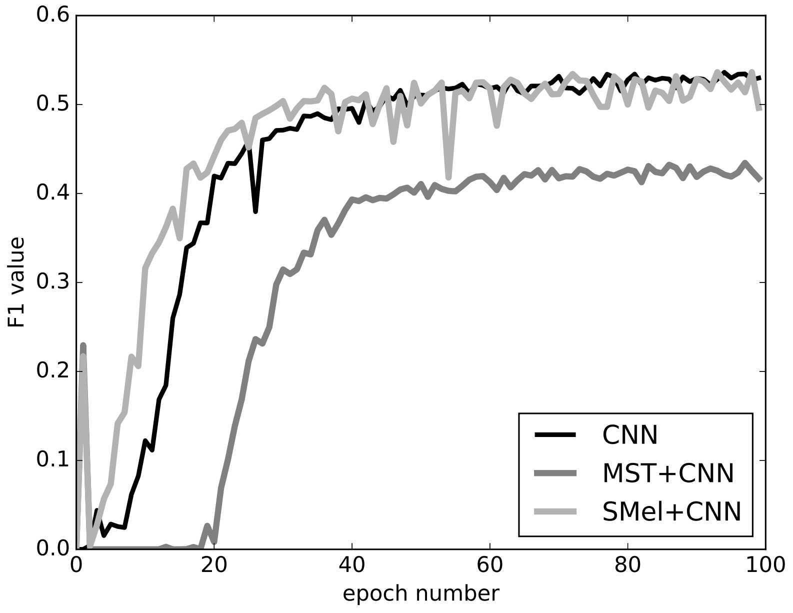


Figure 4.5: Variation of the F1 value in the validation set for each model.

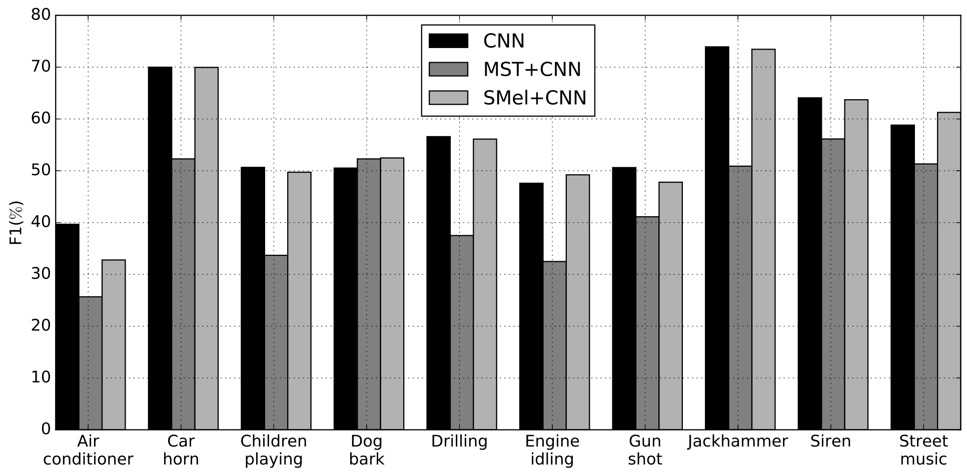


Figure 4.6: F1 values per class on test set for each model.

4.4.3 Energy normalization

In this section, we present the experiments related to PCEN. The SMel model whose outputs are normalized with PCEN is called SMel_P. Firstly, the CNN is trained with the input data also normalized with PCEN using the *librosa* implementation and the same parameters used in Section 2.2.1. This model is called CNN_P. Then, the concatenated network SMel_P+CNN_P is trained

Table 4.1: Results of $F1$ and ER values on the test set (in bold the best results).

Network	$F1(\%)$	ER
CNN	56	0.53
MST+CNN	43	0.61
SMel+CNN	**57**	**0.50**

with the same loss function of equation (4.5). In this experiment, we study how do the filters $H_l[k]$ change, thus w_0 is set to zero to avoid regularization. PCEN parameters are initialized with the same values as in CNN_P. Fig. 4.7 shows the parameter values after the training process. The only parameter that changed significantly is r which defines how the DRC works (see Section 2.2.1). For high frequencies, the r value decreases, and the compression increases (Lostanlen et al., 2019). Analogously, the compression is small for low frequencies. This could be because in this dataset most meaningful information for classification is in the lower frequencies.

Table 4.2: Results of $F1$ and ER on test set for networks CNN, CNN_P and SMel_-P+CNN_P.

Network	$F1(\%)$	ER
CNN	56	0.53
CNN_P	54	0.56
SMel_P+CNN_P	51	0.60

Table 4.2 shows the performance results for CNN, CNN_P and SMel_P+CNN_-P networks. Note that PCEN normalization does not improve the performance of CNN. Also, training SMel_P in conjunction with CNN_P, does not improve the results, but it is interesting to see in the Fig. 4.8 how the $H_l[k]$ filters change.

For high frequencies, the result seems to be very noisy, in particular above 4000 Hz. This suggests that the information in this frequency band does not contribute substantially in the classification. It is interesting to note that this result is similar to the one reported in (Cakır & Virtanen, 2018). To corroborate this finding, we explored re-sampling the dataset at 8000 Hz and calculating the energy on 64 mel bands in the range from 0 to 4000 Hz. Notice that effectively the results of ER and F1 do not change significantly working with a sampling rate of 8000 Hz, while the number of parameters decreases considerably (see Table 4.3). This result could have been obtained by other means, for instance, changing the dataset sampling rate by trial and error. However, the proposed feature extraction network can learn task specific filters when faced to other problems.

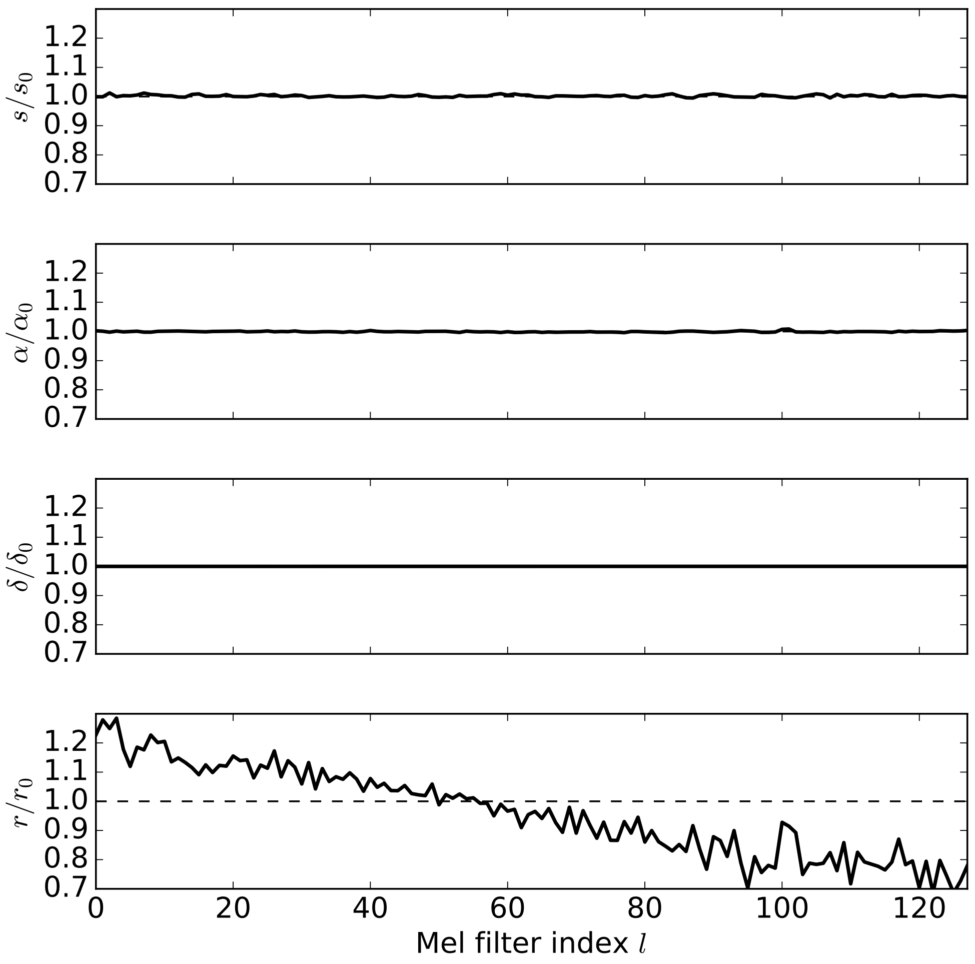


Figure 4.7: PCEN parameter as a function of the channel (band frequency l) referenced to initial values trained with SMel_P+CNN_P. Dash lines mark initial values.

Table 4.3: Results of $F1$ and ER test set and number of parameters of networks CNN_P and SMel_P+CNN_P trained with URBAN-SED dataset re-sampled to 22050 Hz and 8000 Hz.

$f_s(Hz)$	Red	$F1(\%)$	ER	# params (M)
22050	CNN_P	54	0.56	~ 2.48
	SMel_P+CNN_P	51	0.60	~ 2.55
8000	CNN_P	52	0.55	~ 0.99
	SMel_P+CNN_P	49	0.56	~ 1.01

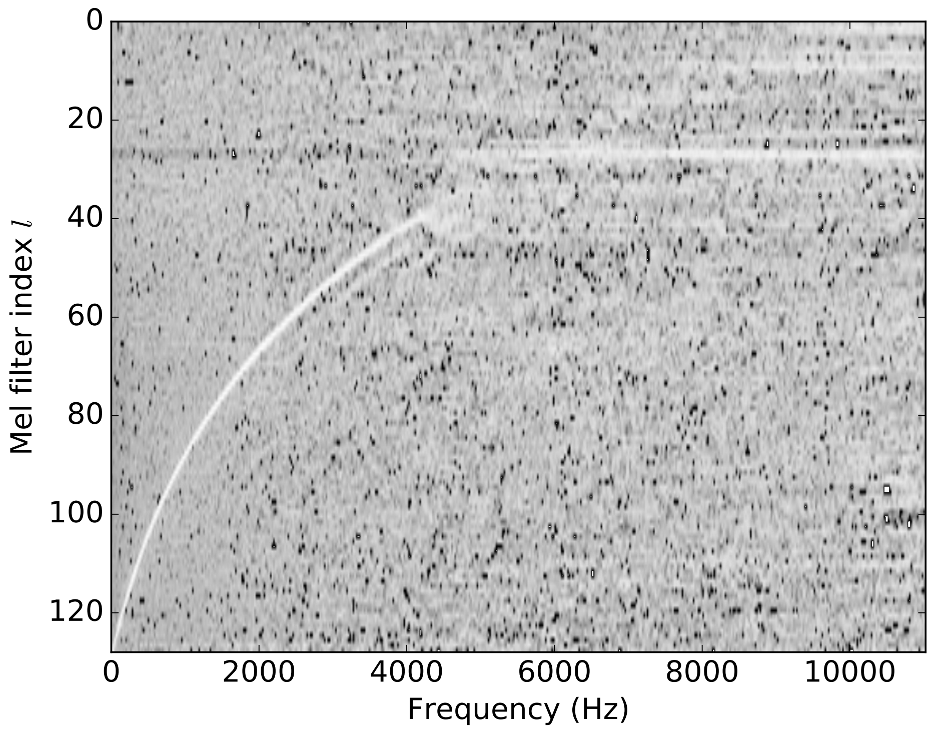


Figure 4.8: $H_l[k]$ filters learned by SMel_P+CNN_P network.

4.5 Discussion

This chapter presents a novel approach for sound event detection in urban environments using end-to-end neural networks, that is obtained by concatenating two networks: one for feature extraction and another one for classification. This two-stage architecture facilitates the introduction of domain knowledge and improves the interpretability of what the networks learn.

For the first network, that is devised to extract the mel-spectrogram, we propose a simple approach based on a mel-frequency filterbank. We show that the proposed model achieves better results (smaller loss function value) than those of the recently proposed MST model. We also show that the classification results of the concatenated end-to-end network are similar to those of a state-of-the-art CNN. However, the proposed model offers a better interpretability regarding the output of the first layers of the network.

Also, we implement the recently proposed PCEN energy normalization as a neural network layer and we train its parameters in conjunction with those of the rest of the network. We find that the only parameter that significantly changes in the training is r, that determines the amount of dynamic range compression of the signal.

We also study how the filters of the first layer change with the training. The results suggest that for the URBAN-SED dataset, the most relevant information is below 4000 Hz and this is confirmed by obtaining similar results for a 8000 Hz sub-sampled version of the dataset. We conclude that models with less parameters could be used.

The main advantage of designing interpretable front-end as the proposed SMel method, is that it can be used for other audio-related tasks. However, the audio representation extracted using SMel, similar to the mel-spectrogram, is a low-level acoustic representation. Therefore interpretable models and backends can be designed to discriminate between classes and extract higher level representations. The following chapters explore this more in depth.

CHAPTER 5

An interpretable model for sound classification

5.1 Introduction

In this chapter, we present a novel explanation-producing neural network for sound classification, with the aim of contributing to the development of interpretable models in the audio domain. The proposed network architecture is based on the interpretable model for image classification introduced in (Li et al., 2018a). The input in our case is a time-frequency representation of the audio signal. The network learns a latent space—by means of an autoencoder—and a small set of prototypes in this latent space. The ability to learn a latent space is the most powerful trait of DNNs and it proved to be a key factor for achieving high performance in the different sound classification tasks addressed. The predictions of the network are based on the similarity of the input to the prototypes in the latent space. We leverage audio domain knowledge to design a frequency-dependent similarity measure and consider different time-frequency resolutions in the feature space, both of which contribute to better discrimination of the sound classes. By applying the decoder function of the autoencoder, a prototype can be mapped from the latent space to the time-frequency input representation and then to an audio signal. The model is constrained to produce good quality audio from the time-frequency representation. This allows for the aural inspection of the learned prototypes and their comparison to the audio input. It is this approach that renders the explainability of the proposed model: the explanation is the set of prototypes—mapped to the audio domain—and the similarity of the input to them.

The conducted experiments show competitive results when compared to that of the state-of-the-art methods in automatic sound classification for three different application scenarios involving speech, music, and environmental audio. Moreover, the ability to inspect the network allows for evaluating its performance beyond the typical accuracy measure. For this reason, some experiments are devised to explore the advantages of the interpretability property of the model. To do so, we propose two automatic methods for network refinement that allow reducing some redundancies, and suggest how the model could be debugged using a human-in-the-loop strategy.

The implemented model and the software needed for reproducing the experiments are available to the community under an open source license from: `https://github.com/pzinemanas/APNet`.

Our main contributions can be summarized, as follows.

1. We propose a novel interpretable deep neural network for automatic sound classification—based on an existing image classification model (Li et al., 2018a)—that provides explanations of its decisions in the form of a similarity measure between the input and a set of learned prototypes in a latent space.

2. We exploit audio domain knowledge to improve the discrimination of the sound classes by designing a frequency-dependent similarity measure and by considering different time-frequency resolutions in the feature space.

3. We rigorously evaluate the proposed model in the context of three different application scenarios involving speech, music, and environmental audio, showing that it achieves comparable results to those of state-of-the-art opaque algorithms.

4. We show that interpretable architectures, such as the one proposed, allow for the inspection, debugging, and refinement of the model. To do that, we present two methods for reducing the number of parameters at no loss in performance, and suggest a human-in-the-loop strategy for model debugging.

The rest of this chapter is organized, as follows. Section 5.2 presents our proposed model. Section 5.3 details the datasets and baseline methods that are used in the experiments reported in Section 5.4. Section 5.5 finalizes the paper with the main conclusions and ideas for future work.

5.2 Proposed Model

The proposed model —called Audio Prototype Network (APNet)—has two main components: an autoencoder and a classifier. The input to the model is a time-frequency representation of the audio signal. The purpose of the autoencoder is to represent the input into a latent space of useful features that are learned during training. The encoded input is then used by the classifier to make a prediction.

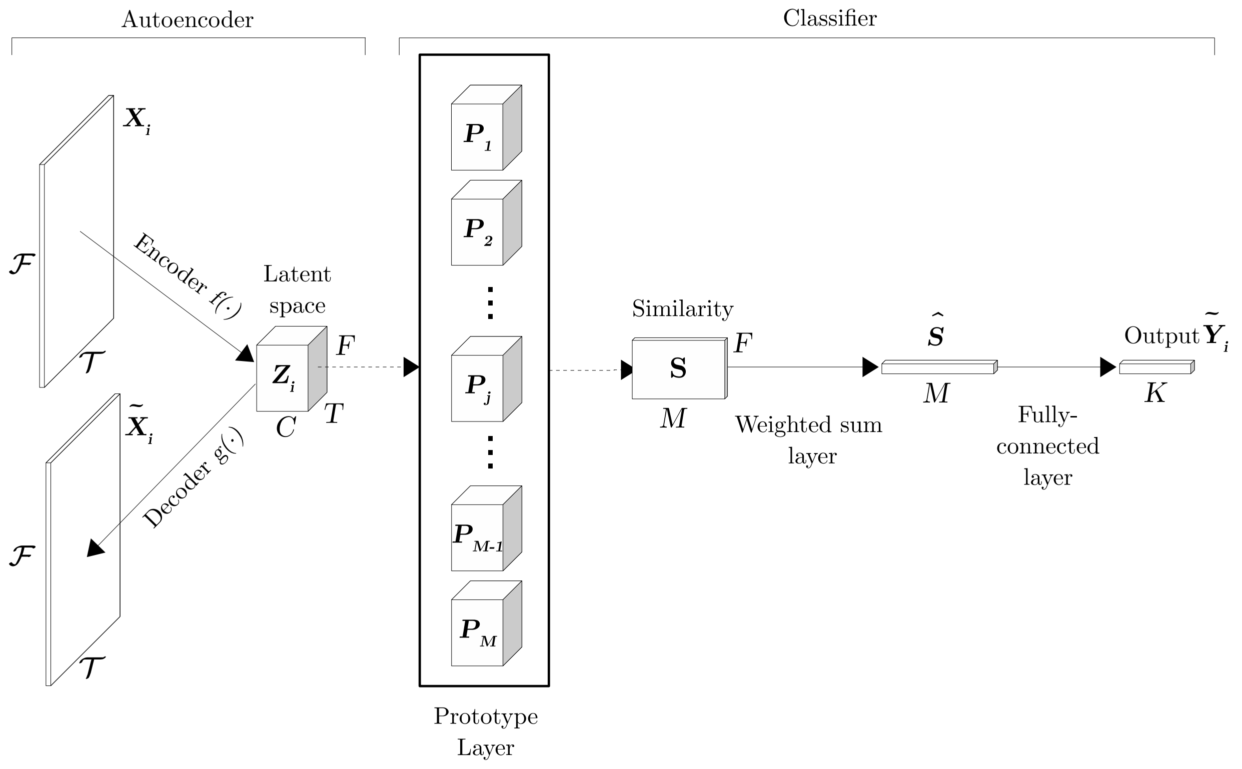


Figure 5.1: Diagram of the proposed model. Continuous arrows represent layers or sub-networks and they have the name below or to the left of them. Dashed line arrows represent unit connections.

The diagram shown in Figure 5.1 represents the network architecture of APNet. The classifier consists of: a prototype layer, a weighted sum layer, and a fully-connected layer. The prediction of the classifier is based on the similarity—computed in the latent space—between the encoded input and a set of prototypes, which are learned during training to be representatives of each class. The weighted sum layer controls the contribution of each frequency bin—in the latent space to—the similarity measure. Finally the fully connected layer takes the weighted similarity measure as input and produces the output prediction. The decoder function of the autoencoder is used to map the prototypes from the latent space to the time-frequency input representation—and then to the audio domain using signal processing. In the following, the main network components are thoroughly described.

5.2.1 Input Representation

The log-scale mel-spectrogram is used as the time-frequency representation of the audio input. This representation is widely used in sound classification, as well as other audio-related problems.

We define the ith input as $\boldsymbol{X}_i \in \mathbb{R}^{\mathcal{T} \times \mathcal{F}}$ where $\mathcal{T}$ and $\mathcal{F}$ are the number of time hops and the frequency bins, respectively. Hence, let $\{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^{N}$ be the training set, where $\boldsymbol{Y}_i \in \mathbb{R}^K$ are the one-hot encoded labels, and N and K are the number of instances and classes, respectively.

5.2.2 Autoencoder

The encoder function $f(\cdot)$ is used to extract meaningful features from the input. Therefore, the encoder transforms the input into its representation in the latent space,

$$\boldsymbol{Z}_i = f(\boldsymbol{X}_i), \tag{5.1}$$

where $\boldsymbol{Z}_i$ is three-dimensional tensor of shape (T, F, C), T and F are the time and frequency dimensions after the encoding operations, and C is the number of channels used in the encoder's last layer. On the other hand, the decoder function, $g(\cdot)$, is devised to reconstruct the input from the latent space,

$$\widetilde{\boldsymbol{X}_i} = g(\boldsymbol{Z}_i) \in \mathbb{R}^{\mathcal{T} \times \mathcal{F}}. \tag{5.2}$$

The autoencoder that is proposed in (Li et al., 2018a) is devised for image processing. We designed the autoencoder of our model to deal with a time-frequency representation as input. In particular, the encoder is suitable for audio feature extraction, and the decoder provides good audio quality in the reconstruction.

Figure 5.2 shows a diagram of the proposed autoencoder. It has three convolutional layers in the encoder and three transpose convolutional layers in the decoder. Besides, we apply max-pooling layers after the first two convolutional layers in order to capture features at different time-frequency resolutions. Note that max-pooling is a non-invertible operation and, thus, there is not an upsampling operation that is suitable for the decoder stage. To overcome this problem, we use the solution that was proposed by Badrinarayanan et al. (2017). This implies saving the indexes used in the max-pooling operations in the form of masks and then using them in the decoder. This mask is set to 1 at the position that maximizes the max-pooling and to 0 otherwise. In the decoder, the mask is applied to the result of a linear upsampling. The next transpose convolutional layer learns how to upsample the masked input.

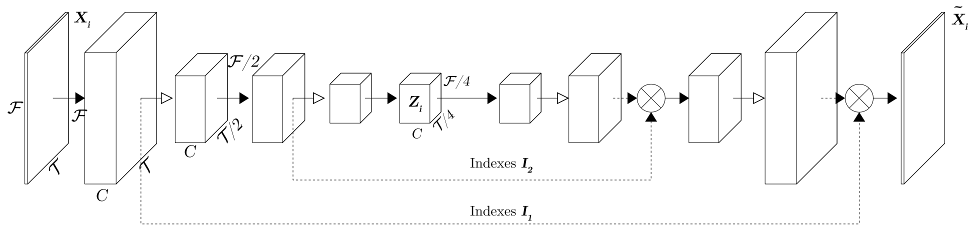


Figure 5.2: Diagram of the proposed autoencoder. Filled arrows represent convolutional (both normal and transpose) layers. The unfilled arrows point out max-pooling (in the encoder) or upsampling (in the decoder) layers. Dashed line arrows illustrate connections.

A leaky ReLu is the activation after each convolutional layer (Maas et al., 2013), except for the last one, which is an hyperbolic tangent in order to limit the reconstruction to the range $[-1, 1]$.

Note that we use padding for all convolutional layers, so that the output has the same shape as the input. Besides, max-pooling operations use a 2×2 window and, therefore, the shape of the encoder's last layer (i.e. the dimension of the latent space) is $(T, F, C) = (\mathcal{T}/4, \mathcal{F}/4, C)$.

For the decoding process to have enough audio quality, it is necessary to optimize the autoencoder by minimizing its reconstruction error. To accomplish this, we use a $L2$ mean square loss function over its inputs and outputs, as

$$l_r = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{X}_i - \widetilde{\boldsymbol{X}_i} \right\|_2^2 . \tag{5.3}$$

5.2.3 Prototype Layer

The prototype layer stores a set of M prototypes that we want to be representatives of each class: $\{\boldsymbol{P}_j\}_{j=1}^{M}$. These prototypes are learned in the latent space and, thus, the shape of $\boldsymbol{P}_j$ is also (T, F, C).

In order to learn the prototypes, we used the same loss function as in the original model (Li et al., 2018a). First, we calculate $\boldsymbol{D} \in \mathbb{R}^{N \times M}$, whose values are the squared $L2$ distance from each data instance to each prototype in the latent space:

$$D_{ij} = \|\boldsymbol{Z}_i - \boldsymbol{P}_j\|_2^2 , \tag{5.4}$$

and then we calculate the following cost function:

$$l_p = \frac{1}{N} \sum_{i=1}^{N} \min_j \{D_{ij}\} + \frac{1}{M} \sum_{j=1}^{M} \min_i \{D_{ij}\}. \tag{5.5}$$

The minimization of this loss function would require that each learned prototype is similar to at least one of the training examples in the latent space; and, vice versa, every training example to be similar to one prototype (Li et al., 2018a). Therefore, training examples will cluster around prototypes in the latent space. Besides, if we choose the decoder to be a continuous function, we should expect that two close instances in the latent space to be decoded as similar instances in the input space, as noted in (Li et al., 2018a). Consequently, we should expect the prototypes to have meaningful decodings in the input space.

The output of this layer is a similarity measure that is based on the distance from each encoded data instance $\boldsymbol{Z}_i$ to each prototype $\boldsymbol{P}_j$, as described in the next section.

5.2.4 Similarity Measure and Weighted Sum Layer

Unlike images, the dimensions of the time-frequency representation have different meanings and they should be treated differently (Pons et al., 2016). For this reason, we propose a frequency-dependent similarity that assigns a different weight to each frequency bin in the latent space. This allows for the comparison of inputs and prototypes to be based on those frequency bins that are more relevant, for instance, where the energy is concentrated.

The similarity measure computation has two steps: (1) the calculation of a frequency-dependent similarity and (2) the integration of the frequency dimension by means of a learnable weighted sum.

The frequency dependent similarity is calculated as a squared $L2$ distance, followed by a Gaussian function. Therefore, the output of the prototype layer, $\boldsymbol{S}$, is obtained by:

$$S_{ij}[f] = \exp\left(- \sum_{t=1}^{T} \sum_{c=1}^{C} (Z_i[t,f,c] - P_j[t,f,c])^2 \right). \tag{5.6}$$

Note that $\boldsymbol{S}$ is a three-dimensional tensor whose shape is (N, M, F), and that we use both sub-indexes and brackets to denote tensor dimensions. Sub-indexes denote the i-th data instance and the j-th prototype; and, brackets denotes the time, frequency, and channels dimensions.

Subsequently, the frequency dimension is integrated, to obtain the matrix $\widehat{\boldsymbol{S}} \in \mathbb{R}^{N\times M}$, while using the following weighted sum:

$$\widehat{S}_{ij} = \sum_{f=1}^{F} H_j[f] S_{ij}[f] \tag{5.7}$$

where $\boldsymbol{H} = \{H_j[f]\} \in \mathbb{R}^{M\times F}$ is a trainable kernel. Note that this is similar to a dot product with a vector of length F for each prototype. We initialize $\boldsymbol{H}$ with all values equal to $1/F$ (equal weight, mean operation), but we let the network learn the best way to weight each frequency bin for each prototype. The kernel is particularly useful to discriminate between overlapping sound classes by focusing on the most relevant frequency bins for each prototype.

5.2.5 Fully-Connected Layer

The fully-connected layer is devised to learn the decisions to transform the similarity measure $\widehat{\boldsymbol{S}}$ into the predictions. Given that the network is intended for a classification task, we use softmax as the activation of this layer. Therefore, the predictions are calculated as:

$$\widetilde{\boldsymbol{Y}} = \mathrm{softmax}(\widehat{\boldsymbol{S}} \cdot \boldsymbol{W}), \tag{5.8}$$

where $\boldsymbol{W} \in \mathbb{R}^{M\times K}$ is the kernel of the layer and $\widetilde{\boldsymbol{Y}} = \{\widetilde{Y}_{ik}\} \in \mathbb{R}^{N\times K}$. Note that we do not use bias in order to obtain more interpretable kernel weights. We expect that for a given output class, the network gives more weight to the prototypes related to this class. For instance, in a problem with K classes and one prototype per class ($M = K$), we expect to learn a fully-connected layer with a kernel close to the identity matrix, $\boldsymbol{W} = \boldsymbol{I}_K$ (Li et al., 2018a). The loss function to train this layer is a categorical cross-entropy,

$$l_c = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} Y_{ik} \log \widetilde{Y}_{ik}. \tag{5.9}$$

Finally, note that, since the prototypes can be converted from the latent space to the time-frequency input representation by applying the decoder function, $g(\boldsymbol{P}_j) \in \mathbb{R}^{\mathcal{T}\times\mathcal{F}}$ is the mel-spectrogram representation of the j-th prototype. Hence, we can illustrate the prediction process while using the mel-spectrograms of data instances and prototypes, even though this is actually performed in the latent space. Figure 5.3 shows an example of this illustration in the context of a classification task with three classes and using one prototype per class ($M = K = 3$).

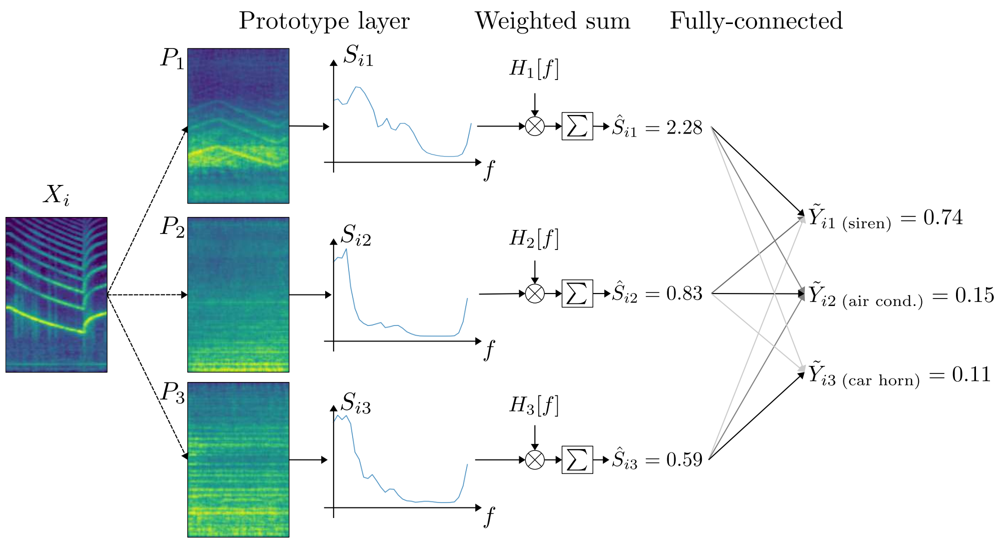


Figure 5.3: Illustration of how the model makes its predictions. This is an example with three classes: *siren*, *air conditioner*, and *car horn*. The input $\boldsymbol{X_i}$ is compared to three prototypes (one per each class) to get the frequency-dependent similarity $\boldsymbol{S}_{ij}[f]$. This similarity is integrated in the frequency dimension using the weighted sum layer to obtain $\widehat{\boldsymbol{S}}_{ij}$. The final step in the reasoning process is to calculate the prediction ($\widetilde{\boldsymbol{Y}}_{ik}$) by projecting the similarity using a fully-connected layer. Note that gray scale in the fully-connected arrows denote the strength of the connection.

5.3 Materials and Methods

In this section, we describe the sound classification tasks addressed, the publicly available datasets, the evaluation methodology, and the baselines that were used for performance comparison.

5.3.1 Sound Classification Tasks

We aim to study the performance of APNet when considering sounds of different nature, such as speech, music, and environmental sounds. To do that, we address three different audio classification tasks: urban sound classification, musical instrument recognition, and keyword spotting in speech.

As show in Chapter 1, in this work, we refer to *sound classification* as the task of assigning a sound category to an audio segment, from a previously defined set of options. We differentiate this from *sound detection*, which also involves locating the sound event within the audio in terms of onset and offset time instant; and, from *audio tagging*, in which multiple labels can be assigned to an audio segment indicating the presence of instances from several sound classes (Mesaros et al., 2016b).

Simple application scenarios have a single sound event per audio segment. This is the case of the keyword spotting task in speech addressed in this work, in which each audio segment has only one short word. Subsequently, a more complex scenario is the presence of a sequence of non-overlapping sound events of the same kind, such as the stem of a single instrument in a multi-track music recording session. The musical instrument recognition task is an example of this kind, despite the fact that some of the instruments (e.g., piano, violin) can simultaneously produce several sounds. The environmental sound scenarios are one of the most complex settings because they typically involve multiple temporally overlapping sound events of different types, and the sounds present often have considerable diversity. This is the kind of problem that is addressed in the urban sound classification task.

Further information regarding the tasks is provided in the following description of the datasets.

5.3.2 Datasets

UrbanSound8k

For the urban sound classification task we use the UrbanSound8K dataset (Salamon et al., 2014). It has more than 8000 audio slices that are extracted from audio recordings from Freesound (Font et al., 2013). Each audio slice is tagged with one of the following ten labels: *air conditioner*, *car horn*, *children playing*, *dog bark*, *drilling*, *engine idling*, *gun shot*, *jackhammer*, *siren*, and *street music*. The length of the audio slices is variable, with a maximum value of 4 s. The audio format is the same as the one originally uploaded to Freesound.

We use the 10-fold cross-validation scheme that was provided by the dataset complying with the following methodology: (1) select a test fold (e.g., fold 1); (2) select a validation set as the next fold (e.g., fold 2); (3) train the model on the rest of the folds (e.g., folds 3 to 10) using the validation set for model selection; and, (4) evaluate the model on the test set. Finally, repeat for the ten folds and average the results.

Medley-Solos-DB

For the music instrument recognition task, we use the Medley-solos-DB dataset (Lostanlen & Cella, 2016). It contains single-instrument samples of nine instruments: *clarinet*, *distorted electric guitar*, *female singer*, *flute*, *piano*, *tenor saxophone*, *trumpet*, and *violin*. Audio clips are three-second length, sampled at 44,100 Hz.

It has a training set extracted from the MedleyDB dataset (Bittner et al., 2014) and a test set from the solosDB dataset (Joder et al., 2009). The training set also includes a split for model validation.

Google Speech Commands

For the keyword spotting task in speech we use the Google Speech Commands V2 dataset (Warden, 2018). It consists of more than 100.000 audio files of one-second length, sampled at 16,000 Hz, each one containing a single short word. Although the dataset is devised for the detection of a small set of commands (e.g., up, down, left, right) and to distinguish them from other short words, we use the complete set of 35 words: *backward*, *bed*, *bird*, *cat*, *dog*, *down*, *eight*, *five*, *follow*, *forward*, *four*, *go*, *happy*, *house*, *learn*, *left*, *marvin*, *nine*, *no*, *off*, *on*, *one*, *right*, *seven*, *sheila*, *six*, *stop*, *three*, *tree*, *two*, *up*, *visual*, *wow*, *yes*, and *zero*. The audio files are organized in three folds for model training, validation, and testing.

5.3.3 Baselines

We compare the performance of APNet to that of three different state-of-the-art opaque models: a Convolutional Neural Network designed for environmental sound recognition in urban environments (SB-CNN) (Salamon & Bello, 2017); a Convolutional Recurrent Neural Network with an attention mechanism devised for speech classification (Att-CRNN) (de Andrade et al., 2018); and, a pre-trained embedding extractor model based on self-supervised learning of audio-visual data (Openl3) (Cramer et al., 2019).

The SB-CNN model is a neural network that is composed by three convolutional layers followed by two dense layers. The Att-CRNN model consists of two horizontal convolutional layers, two bidirectional Long Short-Term Memory (LSTM) recurrent layers, an attention mechanism to integrate temporal dimension, and three dense layers.

Openl3 is a pre-trained embedding extractor model that is based on the self-supervised learning of audio-visual data. The parameters used to calculate the input representation are fixed, and different flavours of the model are available based on the embedding space dimension, the number of mel bands, and the type of data used for training (music or environmental). We use an embedding space of 512 and 256 mel bands for the three datasets. For the urban sound classification task, we select the model trained on environmental data, and the model trained on music for the other two tasks. We use Openl3 as a feature extractor and train a Multi-layer Perceptron (MLP) for each classification task. This network has two hidden layers of 512 and 128 units, respectively, with ReLu activation and dropout with rate 0.5. The dimension of the last layer corresponds to the number of classes in each case.

The three baseline models and APNet use log-scaled mel-spectrograms representation as input, but with different set of parameters. These parameters are also dataset dependent and, therefore, we summarize all combinations in Table 5.1. For instance, in the case of APNet trained on UrbanSound8K, we extract the time-frequency representation with $\mathcal{F} = 128$ bands from 0 to 11,025 Hz, while using a spectrogram calculated with a window length of 4096 and hop size of 1024. Each input corresponds to a 4-seconds slice of the audio signal; hence, the number of time hops is $\mathcal{T} = 84$.

Table 5.1: Mel-spectrogram parameters for each model and dataset: sampling rate in kHz (f_s); window length (w) and hop size in samples (h); number of mel-bands (m); and, audio slice length in seconds (l).

	UrbanSound8K				**Medley-Solos-DB**			
	$\boldsymbol{fs}$	$\boldsymbol{m}$	$\boldsymbol{w, h}$	$\boldsymbol{l}$	$\boldsymbol{fs}$	$\boldsymbol{m}$	$\boldsymbol{w, h}$	$\boldsymbol{l}$
APNet	22.05	128	4096, 1024	4.0	44.1	256	4096, 1024	3.0
SB-CNN	44.1	128	1024, 1024	3.0	44.1	256	1024, 1024	3.0
Att-CRNN	44.1	128	1024, 1024	3.0	44.1	256	1024, 1024	3.0
Openl3	48.0	256	512, 242	1.0	48.0	256	512, 242	1.0

	Google Speech Commands			
	$\boldsymbol{fs}$	$\boldsymbol{m}$	$\boldsymbol{w, h}$	$\boldsymbol{l}$
APNet	16.0	80	1024, 256	1.0
SB-CNN	16.0	80	1024, 256	1.0
Att-CRNN	16.0	80	1024, 256	1.0
Openl3	48.0	256	512, 242	1.0

5.3.4 Training

We train APNet and the baseline models from scratch, without the use of pre-trained weights, except for feature extraction in Openl3. For the three baselines, we optimize a categorical cross-entropy loss over the predictions and the ground-truth, as shown in Equation (5.9). While training APNet, we optimize the weighted sum of all function losses defined previously, Equations (5.3), (5.5), and (5.9):

$$l = \alpha l_c + \beta l_p + \gamma l_r \tag{5.10}$$

where the weights (α, β, and γ) are real-valued hyperparameters that adjust the ratios between the terms. Other important hyperparameters are the number of channels used in the convolutional layers C, the number of prototypes M, and the batch size B. See Table 5.2 for the values of the hyperparameters for each dataset. We train APNet and the baseline models while using an Adam optimizer with a learning rate of 0.001 for 200 epochs. We select the network weights that maximize the accuracy in the validation set.

5.4 Experiments and Results

First, we evaluate the performance accuracy of the proposed model in the three classification tasks considered, and then compare it with that of the baseline

Table 5.2: APNet hyperparameters for each dataset: number of prototypes (M), number of channels in convolutional layers (C); training hyperparameters (α, β, and γ); and, batch size (B).

	$\boldsymbol{M}$	$\boldsymbol{C}$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$	$\boldsymbol{B}$
UrbanSound8K	50	32	(10,5,5)	256
Medley-solos-DB	40	48	(10,5,5)	96
Google Speech Commands	105	48	(2,1,1)	128

models. All of the experiments were conducted using the DCASE-models library (Zinemanas et al., 2020). This simplifies the process of reproducing the experiments, as well as the development of the proposed model. After training and validating all of the models, we evaluate them on the test sets of each dataset. Table 5.3 shows the performance results and the number of parameters of APNet and the three baselines for the three sound classification tasks. These results show that APNet is a very competitive algorithm in all of the tasks, with accuracy values that are comparable to that of the baseline models.

Table 5.3: Mean accuracy (%) and number of parameters (millions) for each model and dataset.

	UrbanSound8K		**Medley-Solos-DB**	
	Acc. (%)	**# Params. (M)**	**Acc. (%)**	**# Params. (M)**
APNet	76.2	1.2	65.8	4.2
SB-CNN	72.2	0.86	64.7	1.8
Att-CRNN	61.1	0.23	52.0	0.29
Openl3	77.3	9.5	67.3	9.5

	Google Speech Commands	
	Acc. (%)	**# Params. (M)**
APNet	89.0	1.8
SB-CNN	92.1	0.17
Att-CRNN	93.2	0.20
Openl3	70.9	9.5

However, unlike the baseline models, APNet is designed to be an interpretable deep neural network. This allows for us to carry out further analysis to provide insight into the inner workings of the network. In particular, the following sections show how to inspect the network (Section 5.4.1) and how to refine the network based on the insights provided by the inspection (Section 5.4.2).

5.4.1 Network Inspection

In this section, we examine the functioning of the autoencoder (Section 5.4.1) and inspect the learned weights from the prototype (Section 5.4.1), fully-connected (Section 5.4.1), and weighted sum (Section 5.4.1) layers.

Autoencoder

The autoencoder is devised to extract meaningful features (encoding process) from the input and to transform back the prototypes from the latent space to the mel-spectrogram representation (decoding process). We qualitatively assess the reconstruction of some data instances in order to examine the functioning of the autoencoder. If this reconstruction is not appropriate, then we can not guarantee that the learned prototypes can be transformed back to the input space in a meaningful way. Figure 5.4 shows the reconstruction of five random signals from one of the validation sets of the UrbanSound8K dataset. The visual comparison of the original and reconstructed mel-spectrogram representations indicates that the decoding process gives adequate results.

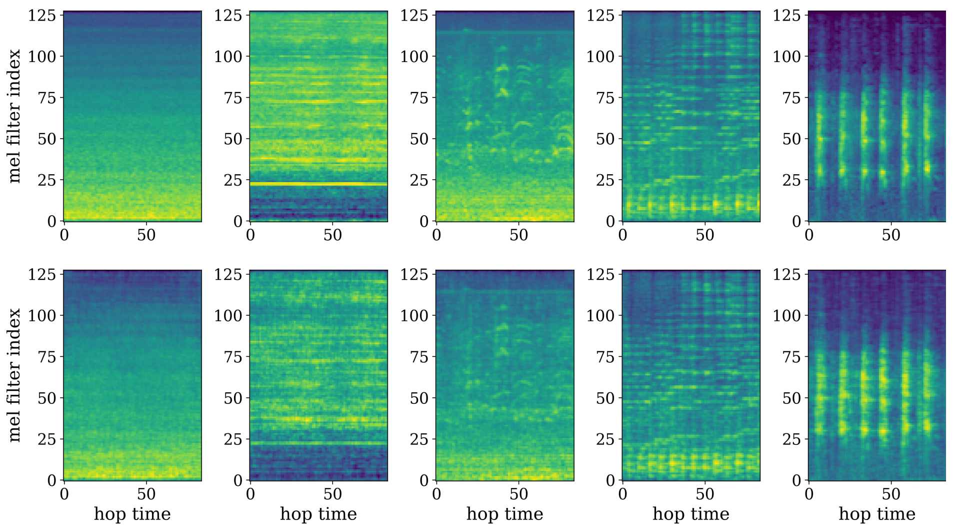


Figure 5.4: Qualitative assessment of the autoencoder. Random instances from the validation set of the UrbanSound8K dataset in the top row ($\boldsymbol{X}_i$) and their respective reconstructions in the bottom row ($\widetilde{\boldsymbol{X}_i}$).

This is also confirmed by listening to the audio signal that is obtained from transforming the time-frequency representation to the waveform domain. The mapping to the audio domain is done by first converting the mel-spectrogram to an approximate linear-frequency spectrogram, followed by the Griffin-Lim (Griffin & Lim, 1984) method. This process is implemented while using librosa (McFee et al., 2015). The audio signals of the examples of Figure 5.4 are available in `https://pzinemanas.github.io/APNet/`.

Prototypes

We take advantage of the decoder to obtain the reconstructed mel-spectrogram of the prototypes, $g(\boldsymbol{P_j})$. Note that, to do that, we need the masks of the max-pooling operations from the encoder function $f(\cdot)$. However, these masks are not available since the prototypes were not transformed by the encoder. However, we can use the masks produced by the instances from the training data that minimize the distance to each prototype. Given that the prototypes are learned to be similar to instances from the training data, we assume that these masks are also suitable for the prototypes. Subsequently, we input these reconstructed mel-spectrograms to the model and obtain the class prediction for each prototype, in order to associate each prototype to a sound class.

Figure 5.5 shows one prototype of each class for the UrbanSound8K dataset. It can be noticed that the mel-spectrograms exhibit the typical traits of the sound classes that they represent.

We also extract the audio signal from each prototype. By listening to the reconstructed audio, one can confirm that the prototypes actually represent their corresponding sound classes. The audio signals of some of these prototypes can be found in `https://pzinemanas.github.io/APNet/`.

Fully-Connected Layer

The fully-connected layer transforms the similarity measure $\hat{\boldsymbol{S}}$ into the network predictions. Recall that the weight matrix $\boldsymbol{W}$ of this layer is designed to be learnable. The analysis of the learned weights of $\boldsymbol{W}$ contributes to the interpretability of the network. More specifically, we are able to tell, from the value of the learned weights, which prototypes are more representative of which class.

Figure 5.6 shows the transposed weight matrix, $\boldsymbol{W}^T$, obtained for the UrbanSound8K dataset. The prototypes are sorted by the prediction that is given by the classifier. Note that most of the prototypes are connected to the corresponding class, with only two exceptions that are related to acoustic similarities. For instance, the last prototype of *air conditioner* is connected to *engine idling*. Besides, there is a strong connection between the first prototype of *dog bark* and the output of *children playing*.

Weighted Sum Layer

The weighted sum layer allows for the network to learn the best way to weight each frequency bin in the latent space for each prototype. It can be useful to focus on the bins where the energy is concentrated or to better discriminate between overlapping sound classes.

Figure 5.7 shows three examples of the learned weights from UrbanSound8K. For instance, in the case of *siren* and *engine idling* prototypes, the weights are

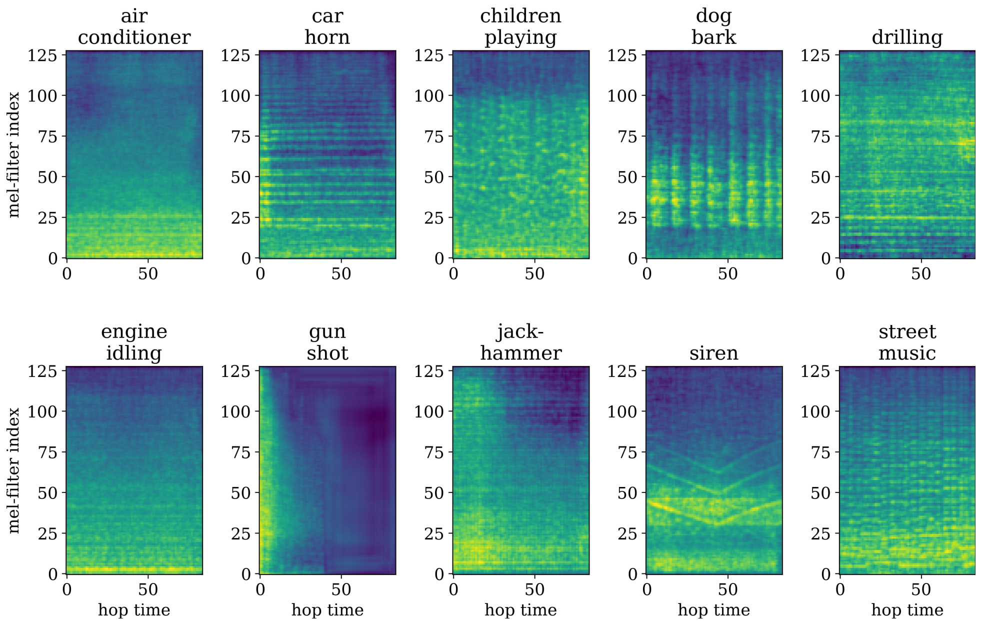


Figure 5.5: Reconstructed mel-spectrograms of selected prototypes, $g(\boldsymbol{P}_j)$.

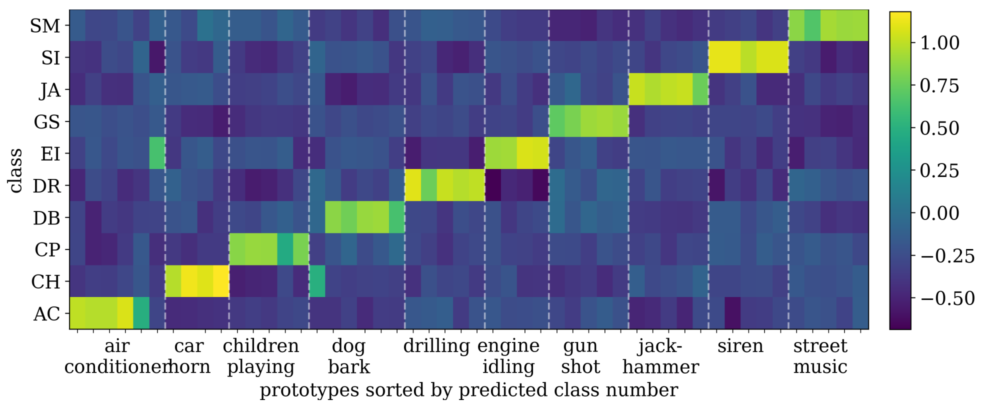


Figure 5.6: Transposed weight matrix of last fully-connected layer, $\boldsymbol{W}^T$.

very correlated to the energy envelope; the layer gives more importance to the low frequencies for *engine idling* and middle frequencies for *siren*. However, in the case of *jackhammer*, this layer gives greater significance to high frequencies, even though the low frequencies are energy predominant. This can be explained as a way to distinguish this class from others with high energy in low frequencies, such as *engine idling* or *air conditioner*.

To show the importance of the trainable weighted sum layer, we undertake the following experiment. We select a data instance that is difficult to classify, because it has a mix of sound sources (a siren, a women yelling, and an engine) and the only tagged class (*siren*) is in the background. We extract the closest prototypes in two cases: (1) using the learned weighted sum; and, (2) repla-

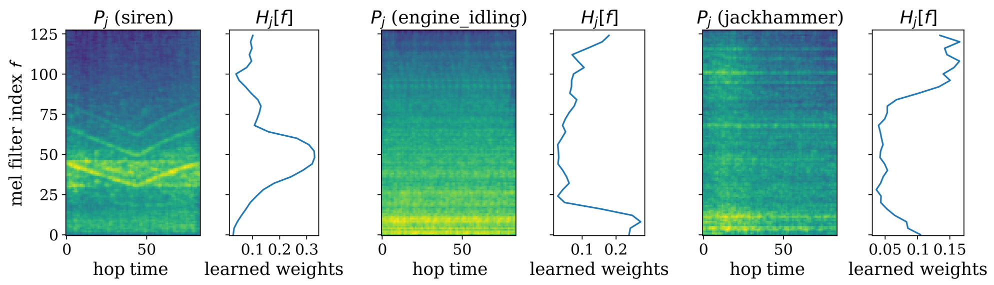


Figure 5.7: Examples of the learned kernel, $\boldsymbol{H}$, of the weighted sum layer from the similarity function. Note that the weights for each prototype $H_j[f]$ are upsampled to compare them in the input space.

cing the weighted sum layer with a fixed mean operation ($H_j[f] = 1/F\ \forall j, f$). Figure 5.8 shows the mel-spectrogram of the data instance ($\boldsymbol{X}_i$) to the left, while, to the right, the three closest prototypes using the mean operation are depicted in the top row, and the three closest prototypes using the trainable weighted sum are depicted in the bottom row. Note that the first two prototypes of the top row correspond to *children playing*—because the predominant source present is a women yelling—and only the third prototype corresponds to the correct class. On the other hand, when the trainable weighted sum is used, the similarity measure is able to capture the source from the background by given more weight to the frequency bins where its energy is concentrated. Note that the first two prototype in the bottom row correspond to the correct class.

5.4.2 Network Refinement

The architecture of APNet allows designers to refine, debug, and improve the model. In the following, we propose two automatic methods to refine the network by reducing redundancy in the prototypes (Section 5.4.2) and the channels in the encoder's last layer (Section 5.4.2). Besides, we present a web application that is devised for the manual editing of the model (Section 5.4.2).

Prototype Redundancy

APNet does not include a constraint on the diversity of the prototypes. As a result, some of the prototypes can be very similar, producing some kind of redundancy. To evaluate this, we calculate the distance matrix of the prototypes $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{M\times M}$, while using the $L2$ squared distance:

$$\mathcal{D}_{jl} = \|\boldsymbol{P}_j - \boldsymbol{P}_l\|_2^2 \,. \tag{5.11}$$

Figure 5.9 shows this distance matrix in the case of the Medley-solos-DB dataset. Note that some of the prototypes of the same class are very similar. To

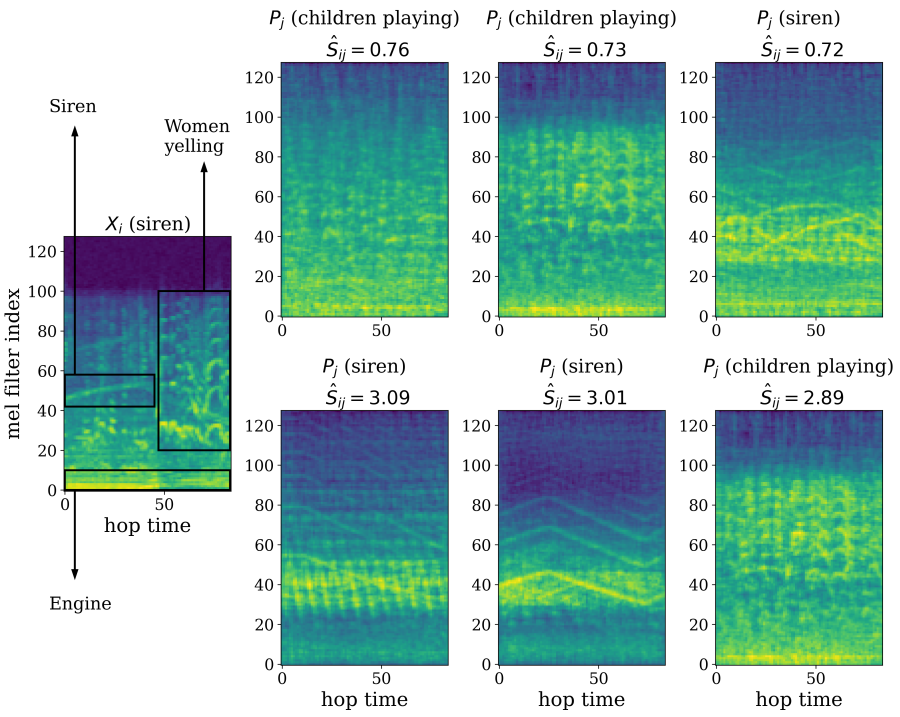


Figure 5.8: Example on the importance of the weighted sum layer. To the left, the mel-spectrogram, $\boldsymbol{X}_i$, of an audio slice that includes several sources: siren in the background along with a women yelling, and an engine in the foreground. Using a mean operation instead of the weighted sum, the two closest prototypes are from *children playing* (**top row**). However, applying the weighted sum, the two closest prototypes are from the correct class, *siren* (**bottom row**).

reduce this redundancy and, consequently, make the network smaller, we remove prototypes that are very close to each other. Formally, we eliminate one prototype for each pair (j, l) that meet:

$$\mathcal{D}_{jl} < \min\{\mathcal{D}_{jl}\} + \text{mean}\{\mathcal{D}_{jl}\}/2 \quad \forall j, l \in [1, \dots, M] : j \neq l \tag{5.12}$$

Note that eliminating these prototypes also implies removing the rows of $\boldsymbol{W}$ associated with them. For instance, if $\boldsymbol{P}_j$ is deleted, then row j from $\boldsymbol{W}$ should also be removed.

After eliminating the redundant prototypes, we train the networks for 20 more epochs using the same parameters from Section 5.4. After this network pruning process, we improve the classification results from 65.8% to 68.2%; and, reduce the number of parameters.

Channel Redundancy

The number of channels of convolutional layers within the autoencoder are usually selected by a grid search method or relying on values used by previous

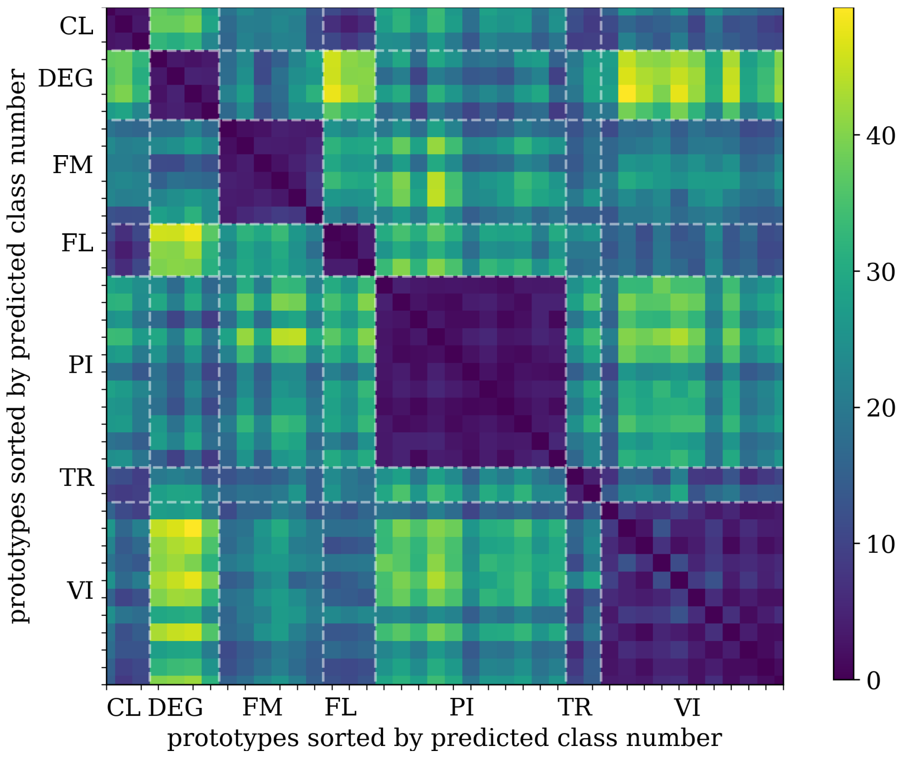


Figure 5.9: Distance matrix of prototypes of the Medley-solos-DB dataset sorted by predicted classes. The labels are *clarinet* (CL), *distorted electric guitar* (DEG), *female singer* (FM), *flute* (FL), *piano* (PI), *trumpet* (TR), and *violin* (VI).

research. A high number of channels can produce network overfitting or add noisy information in the feature space. We take advantage of the prototype architecture to see whether some channels are redundant. Because the output of the last convolutional layer of the encoder yields the latent space, we can use the prototypes to analyze the channel's dimension. Therefore, if there is redundant or noisy information in the prototypes, this can be related to the filters in the last layer of the encoder.

To study this, we undertake the following experiment in the instrument recognition task. We calculate the accumulated distance matrix, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{C\times C}$, of the prototypes as a function of the channel dimension:

$$\mathcal{C}_{kq} = \sum_{i,j,t,f} \left(P_i[t,f,k] - P_j[t,f,q]\right)^2 . \tag{5.13}$$

Subsequently, we find the eight minimum values of $\mathcal{C}$ outside the diagonal and we delete one channel from each pair (k,q). We delete all of the weights related to these channels and, as a result, we reduce the number of filters of the last layer from 48 to 40. We re-train the network for 20 more epochs, because the decoder part has to be trained again.

Table 5.4 shows the results that were obtained for the Medley-solos-DB dataset after both prototype and channel refinement processes, when compared to Openl3 as the most competitive baseline. After the two refinements the accuracy increases by 3.3%, while the number of parameters is reduced by 2.6 M. Besides, this result improves the Openl3 performance by 1.8%.

Table 5.4: The accuracy and number of parameters for APNet before and after the refinement processes for the Medley-solos-DB dataset. Openl3 is included as the most competitive baseline.

Network	**Accuracy** (%)	**Approx. # Parameters**
APNet	65.8	4.2 M
APNet (R. prototypes)	68.2	1.8 M
APNet (R. channels)	69.1	1.5 M
Openl3	67.3	9.4 M

Manual Editing

In the previous sections, we described two automatic methods to refine APNet that lead to better performance and smaller networks. However, when working with networks designed for interpretability like APNet, it is also possible to visualize the models and allow the users to refine them manually. To that end, we designed a web application that allows for the users to interact with the model, refine it, and re-train it to obtain an updated model. Figure 5.10 shows a screenshot of the web application. The user can navigate in a two-dimensional (2D) representation of the prototypes and the training data. It is also possible to see the mel-spectrograms of prototypes and the data instances and to play the audio signals. The user can remove selected prototypes and convert instances to prototypes. Once the prototypes are changed, the model can be re-trained and evaluated.

Using this tool, manual debugging of the model is also possible. Furthermore, different training schemes, such as human-in-the-loop strategies, are possible. For instance, the user can periodically check the training and change the model after a few numbers of epochs. It is also a good starting point to design interfaces for explaining the inner workings of deep neural networks to end-users.

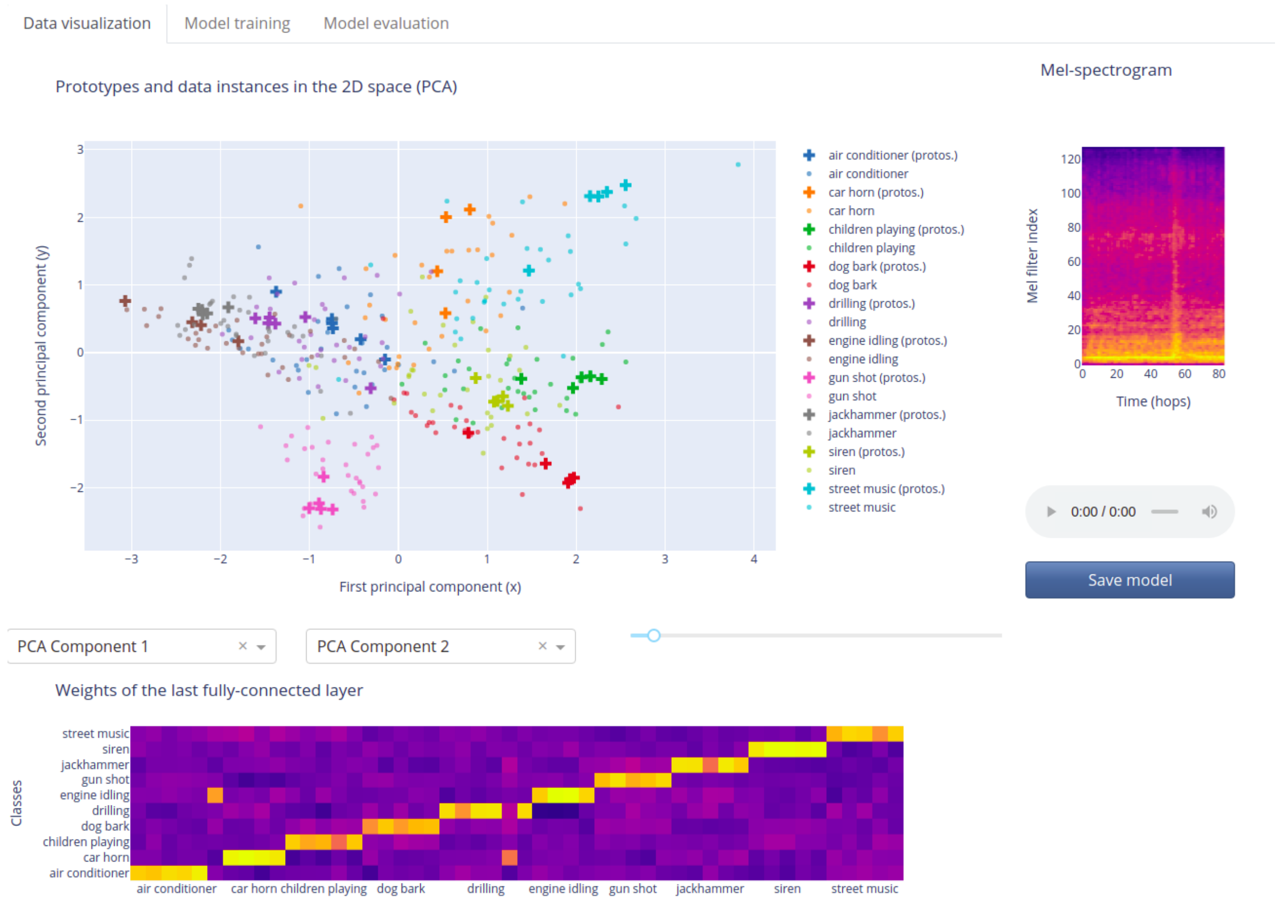


Figure 5.10: Screenshot of the web application designed for the manual editing of APNet. In this tab the user can navigate through the training set and the prototypes in a two-dimensional (2D) space, listen to the audio files and explore the mel-spectrograms. It is also possible to delete prototypes and convert data instances into prototypes. Other tabs include functions for training and evaluating the edited model.

5.5 Discussion

In this chapter, we present a novel interpretable deep neural network for sound classification—based on an existing model devised for image classification (Li et al., 2018a)—which provides explanations of it decisions in terms of a set of learned prototypes in a latent space and the similarity of the input to them.

We leverage domain knowledge to tailor our model to audio-related problems. In particular, we propose a similarity measure that is based on a trainable weighted sum of a frequency-dependent distance in a latent space. Our experiments show that including the trainable weighted sum effectively improves the model, in particular when classifying input data containing mixed sound sources.

The proposed model achieves accuracy results that are comparable to that of state-of-the-art baseline systems in three different sound classification tasks: urban sound classification, musical instrument recognition, and keyword spotting in speech.

In addition, the ability to inspect the network allows for evaluating its performance beyond the typical accuracy measure and provide useful insights into the inner workings of the model. We argue that the interpretability of the model and its reliable explanations—in the form of a set of prototypes and the similarity of the input to them—increase its trustworthiness. This is important for end-users relying on the network output for actionable decisions, even in low-risk applications.

The interpretable architecture of APNet allows designers to refine, debug, and improve the model. In this regard, we propose two automatic methods for network refinement that eliminate redundant prototypes and channels. We show that, after these refinement processes, the model improves the results, even outperforming the most competitive baseline in one of the tasks. Our results exemplify that interpretability may also help to design better models. This contrasts with the widely extended assumption that there is an unavoidable trade-off between interpretability and accuracy.

A limitation of this model is that the prototype loss strongly regularize the ts latent space. For instance, the prototypes of different classes tend to be separated to perform sound classification correctly. However, in a polyphonic setting, i.e., with several sources sounding simultaneous, an instance has to be close to several prototypes simultaneously. This is not possible in such regularized latent space. In Chapter 6, we explore how to extend the approach presented in this chapter to the polyphonic setting, such as the sound event detection task.

Future work includes using the visualization tool for manual editing to study different ways of training the network with a human-in-the-loop approach and creating tools for explaining the inner functionality of the network to end-users.

We explore these research lines more in-depth in Chapter 7.

CHAPTER 6

An extension for sound event detection

6.1 Introduction

In the Chapter 5, we proposed a prototype-based method for audio classification, namely Audio Prototype Network (APNet). This model showed compelling results when applied to speech, music, and environmental audio, for problems with a single class label per audio clip. However, in a polyphonic setting (i.e multi-label), an input instance corresponding to several classes should be simultaneously close to prototypes of those classes in the latent space. Unfortunately, learning such latent space proved challenging in practice, thus motivating the alternative approach proposed in this chapter.

In this chapter, we propose to extend this method to tackle the problem of interpretable polyphonic sound event detection. To provide interpretability, we leverage the prototypes network approach and attention mechanisms. The network learns *local prototypes*, i.e. data points in the latent space representing a patch in the input representation. The approach is similar to that of (Chen et al., 2019a) for single class image classification, which compares image parts to learned prototypes. However, we extend the scope to a multi-label setting with promising results in sound event detection. Besides, the proposed model learns attention maps used for positioning the local prototypes and reconstructing the latent space properly. Then, the detection is solely based on the attention maps. Thus, the explanations of the network are in the form of local prototypes and attention maps.

6.2 Proposed model

Let $\boldsymbol{X}_i \in \mathbb{R}^{\mathcal{T}\times\mathcal{F}}$ be the i-th mel-spectrogram where $\mathcal{T}$ and $\mathcal{F}$ are the number of time frames and frequency bins, respectively. Therefore we define the training set as $\{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^{N}$, where $\boldsymbol{Y}_i \in \mathbb{R}^K$ are the one-hot encoded labels, N is the number of instances and K is the number of classes. APNet is formed by two main components: an autoencoder and a classifier (see Section 5.2). The proposed model uses the same autoencoder from APNet, which is represented in the upper branch of Figure 6.1, and utilizes a novel classifier. The encoder is aimed at extracting meaningful features from the input:

$$\boldsymbol{Z}_i = f(\boldsymbol{X}_i), \tag{6.1}$$

where $\boldsymbol{Z}_i$ is a tensor of shape (T, F, C) and represents the transformed input in the latent space. C is the number of channels of the encoder's last convolutional layer. The decoder part of the autoencoder is used for reconstructing the mel-spectrogram:

$$\widetilde{\boldsymbol{X}_i} = g(\boldsymbol{Z}_i) \in \mathbb{R}^{\mathcal{T}\times\mathcal{F}}. \tag{6.2}$$

Both the encoder and the decoder are formed by three convolutional layers with leaky ReLu activations. The encoder includes two max-pooling layers interspersed between the convolutions and the decoder applies the corresponding unpooling layers. Please refer to Section 5.2 for more details. The classifier of APNet is based on the distance from $\boldsymbol{Z}_i$ to a set of M prototypes with the same shape (T, F, C). Therefore, a prototype is a point in the latent space corresponding to the full mel-spectrogram representation in the input space. This makes it troublesome for APNet to represent a multi-label input instance as it should be close to prototypes from different classes.

The model proposed in this work is devised to overcome this limitation, i.e. it is capable of detecting various simultaneous sound events. The middle and bottom branches in the diagram of Figure 6.1 show this novel classifier. We use another encoder, $s(\cdot)$, to extract M attention maps in the latent space:

$$\boldsymbol{S}_i = s(\boldsymbol{X}_i), \tag{6.3}$$

where $\boldsymbol{S}_i$ is a tensor of shape (T, F, M). The encoder $s(\cdot)$ is similar to the autoencoder's one, $f(\cdot)$, but with ReLu activations to force a non-negative output. Each attention map is related to one prototype. Therefore the network learns a set of M prototypes of shape $(1, 1, C)$. We represent the M prototypes as a tensor $\boldsymbol{P}$ of shape $(1, 1, M, C)$. Note that each prototype represents one point in the time-frequency plane in the latent space. Therefore, in the input space these prototypes represent a patch of shape equal to the receptive field of the encoder network (32×32 in this work).

Using the attention maps and the learnable prototypes the model tries to reconstruct the latent representation $\boldsymbol{Z}_i$. This is done by multiplying each

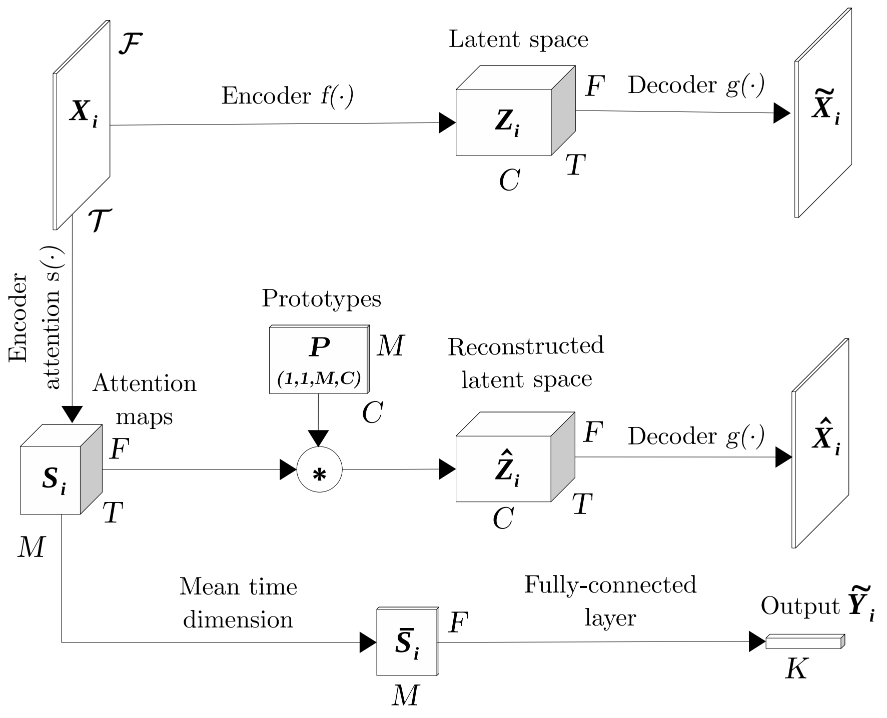


Figure 6.1: Diagram of the proposed model.

attention map by its corresponding prototype and then summing all maps. Note that this is equivalent to a 1×1 2D convolutional layer:

$$\widehat{\boldsymbol{Z}}_i = \boldsymbol{S}_i * \boldsymbol{P}, \tag{6.4}$$

or a dense layer:

$$\widehat{\boldsymbol{Z}}_i = \boldsymbol{S}_i \cdot \boldsymbol{P}_s, \tag{6.5}$$

where $\boldsymbol{P}_s$ is the squeezed version of the tensor $\boldsymbol{P}$ with shape (M, C).

Therefore $\widehat{\boldsymbol{Z}}_i$ has the same shape of $\boldsymbol{Z}_i$ and aims to be a reconstruction of the latent space. In summary the attention maps represent the specific weight of each local prototype in each time-frequency point in order to have a good reconstruction of the latent space. Using the decoder $g(\cdot)$ from the top branch, we can project this reconstructed tensor into the input space,

$$\widehat{\boldsymbol{X}}_i = g(\widehat{\boldsymbol{Z}}_i) \in \mathbb{R}^{\mathcal{T} \times \mathcal{F}}. \tag{6.6}$$

In this way, we can visualize the reconstruction of the latent space in the input space to inspect it.

Finally, the bottom branch deals with the detection task, which is solely based on the attention maps. First, we average the time dimension of $\boldsymbol{S}_i$:

$$\bar{S}_i[f, m] = \frac{1}{T} \sum_{t=1}^{T} S_i[t, f, m] \tag{6.7}$$

where $\bar{\boldsymbol{S}}_i$ has shape (F, M) and integrates the attention map for each prototype and frequency bin in the latent space. Then, a dense layer connects a flattened version of $\bar{\boldsymbol{S}}_i$ with the classification output:

$$\widetilde{\boldsymbol{Y}}_i = \text{sigmoid}(\bar{\boldsymbol{S}}_i \cdot \boldsymbol{W}), \tag{6.8}$$

where $\boldsymbol{W} \in \mathbb{R}^{MF \times K}$ is the kernel of the layer and $\widetilde{\boldsymbol{Y}}_i = \{\widetilde{Y}_{ik}\} \in \mathbb{R}^{1 \times K}$. We do not use bias in order to keep this layer more interpretable. We seek to audit how the model connects each prototype and each frequency bin to the corresponding output.

6.2.1 Objective function

We want the model to be able to detect sound events while maintaining the interpretability of the parameters and the explainability of the predictions. For this purpose, we define three losses to train the model. First we have a loss for learning the detection task. Since this is a multi-label problem, we use binary cross-entropy, $\mathcal{L}_c$. Then we define a mean squared error loss to have good reconstruction quality in the autoencoder of the top branch:

$$\mathcal{L}_r = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{X}_i - \widetilde{\boldsymbol{X}}_i \right\|_2^2. \tag{6.9}$$

This loss ensures that we can transform the data from the latent space back to the input space, in particular the learned prototypes.

Finally, we define a loss for enforcing a correct process of reconstruction using the attention maps and the prototypes. In other words, we let the network learn how to position the prototypes using the attention maps. In this sense, we define a mean squared error loss in both latent and input spaces:

$$\mathcal{L}_p = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{Z}_i - \widehat{\boldsymbol{Z}}_i \right\|_2^2 + \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{X}_i - \widehat{\boldsymbol{X}}_i \right\|_2^2. \tag{6.10}$$

This loss ensures two assets of the model related to its interpretability. First, this loss establishes that the attention maps are learned to be an explicit explanation of how the model makes its predictions. Note that the attention maps are the only information used for the final prediction. And these maps are interpretable since they show how to position each prototype in the latent space in order to have a good reconstruction. Moreover this loss ensures that the prototypes are similar to the data and therefore we can transform them to the input space and audit them.

Besides, we use $l1$ regularization to force some sparsity in the attention map:

$$\mathcal{R}_s = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{Z}_i\|_1. \tag{6.11}$$

This is to prevent the network from reconstructing the latent space by mixing many prototypes. We also apply the same type of regularization to the kernel of the dense layer that connects the attention maps and the output: $\mathcal{R}_w = \|\boldsymbol{W}\|_1$. The idea is that the output for a given class is activated with only a few points on the attention map, both in the frequency and prototype dimension. Therefore we keep the explanations as simple as possible.

While training the proposed system, we optimize the weighted sum of all losses and regularization terms defined previously:

$$\mathcal{L} = \alpha\mathcal{L}_c + \beta\mathcal{L}_r + \gamma\mathcal{L}_p + \delta\mathcal{R}_s + \epsilon\mathcal{R}_w \qquad (6.12)$$

where the weights $(\alpha, \beta, \gamma, \delta, \epsilon)$ are real-valued hyperparameters.

6.3 Experiments and results

We train the proposed model by optimizing the objective function defined in Eq. (6.12). We use Adam optimizer with a learning rate of 0.001 for 50 epochs and we select the model with the top performance in the validation set. We use the following set of hyperparameters $(10, 5, 5, 10^{-5}, 10^{-6})$ and a batch size of 256. The experiments are conducted using the DCASE-models library (Zinemanas et al., 2020) and the code is available under an open-source license[8].

We compare the performance of the proposed model to that of two different opaque baselines: (1) a Convolutional Neural Network (CNN) formed by three convolution layers and two dense layers (Salamon et al., 2017); and (2) a MLP whose input is the embedding vector extracted from the pre-trained Openl3 model (Cramer et al., 2019). We optimize a binary cross-entropy loss with the same optimizer and strategy for these baselines as for the proposed model.

We train and evaluate the proposed model and the baselines on the URBAN-SED dataset v2.0 (Salamon et al., 2017). This is formed by 10-second length audio files corresponding to synthetic mixtures of sound sources obtained from the UrbanSound8k dataset. Each sound event is tagged with one of the following classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. The three models use log-scaled mel-spectrogram as input representation, but with different parameters. Both the CNN and the proposed model uses 128 mel bands and a sampling rate of 22050 Hz. The proposed model uses a window size of 4096 and hop size of 1024 for calculating the spectrograms. On the other hand, CNN uses a window size of 512 and hop size of the same length. Openl3 has predefined parameters (Cramer et al., 2019).

To evaluate the models, we use F-score ($F1$) and error rate Error Rate (ER) in a 1-second grid as commonly used for sound event detection (see Section 2.2.5).

[8] `https://github.com/pzinemanas/attprotos`

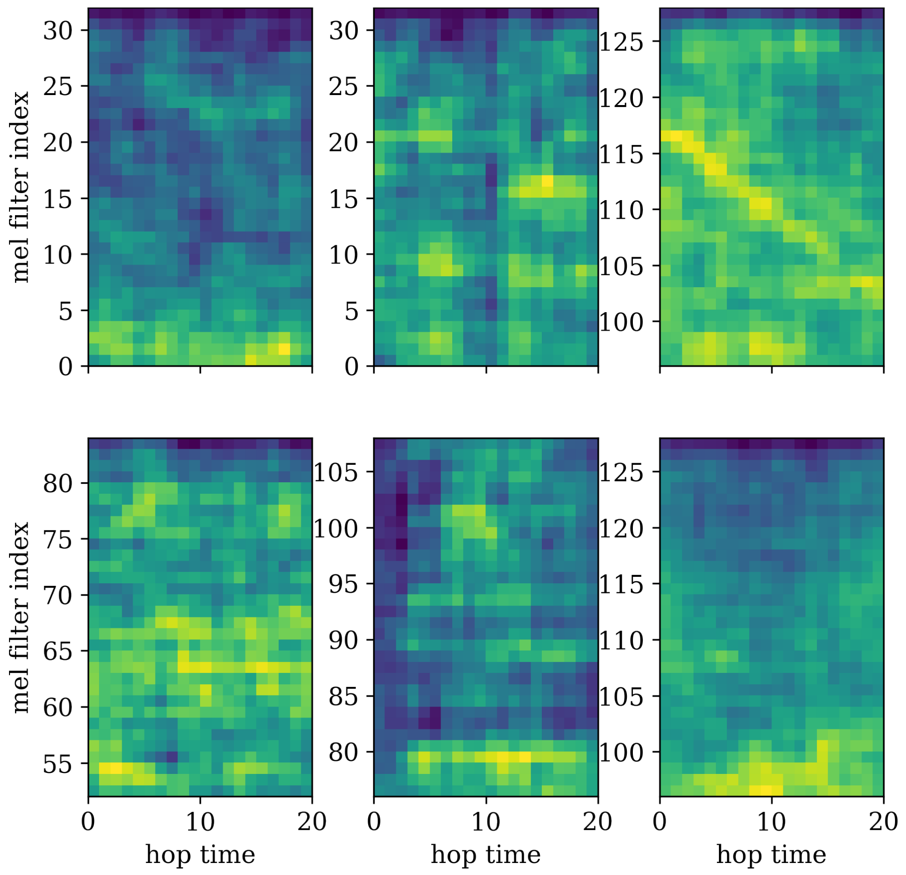


Figure 6.2: Reconstructed learned local prototypes. The y axis represents the mel bands where the prototypes were reconstructed.

We run the training 10 times and calculate the mean and standard deviation of both metrics. Table 6.1 shows the performance comparison of the three models along with their number of parameters. Note that the performance of the proposed model is comparable to that of the baselines, but with fewer parameters.

Table 6.1: Performance comparison of the proposed model and the two baselines. The performance metrics are the F-score ($F1$) and the Error Rate (ER). The number (#) of parameters in millions (M) are also included in the comparison.

Network	$\boldsymbol{F1}$ **(%)**	$\boldsymbol{ER}$	**# Params. (M)**
CNN	57.3 ± 0.6	0.568 ± 0.006	0.5
Openl3+MLP	58.2 ± 0.3	0.558 ± 0.004	9.5
Ours	58.8 ± 0.9	0.572 ± 0.007	0.15

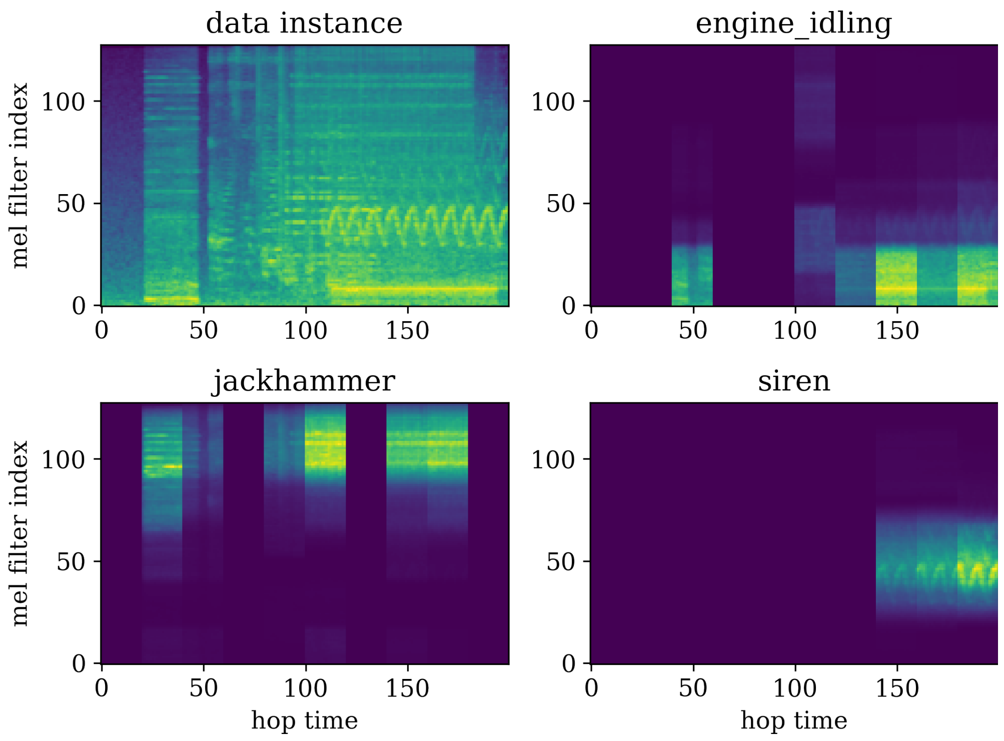


Figure 6.3: Example of an input instance from the test set masked by the attention maps. At the top left plot we show the mel-spectrogram, which includes sound events of six different classes. The other plots are the same mel-spectrogram but masked by each of the reconstructed attention maps for the corresponding classes.

6.3.1 Prototypes

The reconstruction of the latent space helps the network to learn prototypes similar to patches from the training data. We use the decoder part of the autoencoder, $g(\cdot)$, to reconstruct the learned prototypes in the input space. We follow a process similar to that performed in APNet for this purpose (see Section 5.4.1). But in this case, we have to extend the prototypes tensor $\boldsymbol{P}$ to have the same shape of the latent space, i.e. (M, T, F, C). To this end, we create a zero tensor of this shape and select a point in the time-frequency plane where to position each prototype. The time is selected arbitrarily at the center, and the frequency is selected by minimizing the distance of each prototype to the data instances. By doing this, we reconstruct the patches in the frequency bands where the closest data instances have these prototypes present. Figure 6.2 shows a set of selected prototypes. Note that the network learns different types of shapes and textures related to environmental sounds present in the data set.

6.3.2 Attention maps

For each data instance it is possible to extract the corresponding attention maps to provide and explanation on how the model makes its predictions. For a given class k, we follow the following process:

1. Mask the prediction $\widetilde{\boldsymbol{Y}}_i \in \mathbb{R}^{1\times K}$ by a unit vector of the same shape whose k-th component is the only one equal to 1:
$$\widetilde{\boldsymbol{Y}}_i^{(k)} = \widetilde{\boldsymbol{Y}}_i \odot \mathbf{1}_k$$

2. Get the points of the previous layer that are more connected to the output k by calculating the gradient:
$$\nabla \bar{\boldsymbol{S}}_i^{(k)} = \widetilde{\boldsymbol{Y}}_i^{(k)} \cdot \boldsymbol{W}^T \in \mathbb{R}^{1\times FM}$$

3. Reshape the gradient to (F, M), apply a half-wave rectifier to keep only positive connections and multiply it by the time-averaged attention maps:
$$\bar{\boldsymbol{S}}_i^{(k)} = \text{ReLu}\left(\nabla \bar{\boldsymbol{S}}_i^{(k)}\right) \odot \bar{\boldsymbol{S}}_i.$$
This represents the attention maps masked by the most important connections to the output k.

4. Find the most connected prototype by maximizing the energy of the masked attention map:
$$\widehat{m} = \underset{m\in[1,\ldots,M]}{\arg\max} \sum_{f=1}^{F} \left(\bar{S}_i^{(k)}[f,m]\right)^2$$

5. Extract the frequency-dependent attention function:
$$\mathcal{S}_i^{(k)}[f] = \bar{S}_i^{(k)}[f,\widehat{m}]$$

6. Convert the attention function to the input space. To do this, we first upsample the sequence by a rate of 4 to emulate the two max-pooling operations. Then we apply a moving-average filter to emulate the receptive field. Thus, the length of the filter is equal to the receptive field (32).

Figure 6.3 shows an example of the attention maps for three different classes. We multiply the attention maps in the input space by the mel-spectrograms, similarly to how the model does in the latent space. Note that the model can detect simultaneous sound events whose energy is concentrated in different frequency bands. Since the attention maps are designed to reconstruct the latent space and are the only information used for classification, these represent the inherent explanation of how the network makes its predictions.

6.4 Discussion

In this chapter, we present a novel interpretable model for polyphonic sound event detection. Its predictions are based on attention maps learned for reconstructing the latent space by positioning a set of local prototypes. The network also learns the local prototypes as data points in the latent space representing a patch in the input representation. The attention maps provide a form of explanation that is faithful to the model computations and can give valuable insights into its decision process. Moreover, the prototypes can be reconstructed and thus can be listened to and audited.

The proposed model achieves encouraging results in urban sound event detection for a data set of synthetic mixtures, which are comparable to that from two opaque baselines but with fewer parameters, while at the same time offering interpretability. This is consistent with some previous work that claims that it is often possible to incorporate interpretability into deep-learning models to tackle complex tasks without sacrificing performance (Li et al., 2018a; Chen et al., 2019a; Rudin, 2019).

Future work includes ablation studies to understand better the impact of the proposed losses and regularization terms in the final model. In addition, more experiments are needed to evaluate the effect of some hyperparameter values, such as the loss weights and the number of prototypes. Besides, we should evaluate the model with datasets recorded in natural conditions. Finally, we seek further development of interpretable models to analyze environmental sounds, including those that learn disentangled representations.

CHAPTER 7

Conclusions and future perspectives

7.1 Introduction

This thesis has described several interpretable deep-learning models for sound classification and event detection. We have worked on both interpretable audio representations and discriminators. Given that post-hoc methods for interpretability tend to be less reliable (as noted in Chapter 1) in this thesis we focus on intrinsic interpretable methods. We have focused on environmental sound recognition as the downstream task since this is an exciting case study for applying interpretability in the audio domain. Throughout the thesis, we have illustrated how we can use the insights obtained from interpretable methods to give explanations to users, debug and refine models, and understand the training data better. We have also presented tools to foster reproducibility and to contribute to the development of open-source machine learning models in the context of the DCASE community.

We started with an introduction to the motivation of this thesis, interpretable models, and environmental sound recognition systems (Chapter 1). We continued by summarizing the background of this thesis, including the literature review on interpretable deep learning and sound event detection and classification (Chapter 2). We also described DCASE-models, a Python library for improving the reproducibility of deep-learning-based algorithms within the DCASE community (Chapter 3). Then we described an interpretable representation for end-to-end deep-learning systems for sound event detection (Chapter 4). The following two chapters focused on the design of interpretable deep-learning architectures for sound classification (Chapter 5) and sound event detection (Chapter 6).

In this concluding chapter, we first expose the main contributions of this thesis, including those not directly reflected in this document (Section 7.2). We then discuss the strengths and limitations of the intrinsic interpretable methods proposed in this thesis (Section 7.3). Finally, we highlight some possible directions to improve deep-learning models for audio-related problems (Section 7.4).

7.2 Summary of contributions

This thesis has contributed to advancing the state of the art in the audio research community regarding the development of interpretable deep-learning models. In particular, it has contributed to the development of intrinsic interpretable methods for sound classification and sound event detection, mainly in the context of the DCASE community. We also subscribed to open-data and open-science guidelines and made efforts to develop software libraries to promote transparent and reproducible research. The main contributions of this thesis are summarized as follows:

- It comprehensively reviews the environmental sound recognition field, including the most popular machine learning methods, input representations, datasets, and metrics.
- It also provides an extensive review of the state of the art of interpretable deep-learning models, including post-hoc and intrinsic methods. Besides, it reviews the related work on interpretable deep-learning methods for sound event recognition.
- It contributes with software tools (DCASE-models and soundata) and open datasets (MAVD and UrbanSaS) to promote reproducible research in environmental sound recognition.
- It explores the use of interpretable audio representations in the context of end-to-end deep-learning systems for sound event detection.
- It proposes an interpretable deep-learning model for sound classification based on prototypes, including layers devised using domain knowledge. It also illustrates the use of this framework to design human-in-the-loop strategies, refining and debugging the model.
- It describes an extension of the interpretable deep-learning model for sound classification to deal with polyphonic sound event detection. This model takes advantage of the prototypes to describe the local representation of the data and attention maps to locate those prototypes.

The research carried out in this thesis gave rise to several publications. We published the description of the DCASE-models library (Chapter 3) in the context of the DCASE conference (Zinemanas et al., 2020). In addition, we published the outcomes from Chapter 4 in a conference paper (Zinemanas et al., 2019a). Furthermore, we published the outcomes from Chapters 5 and 6 in a journal paper (Zinemanas et al., 2021b) and a conference paper (Zinemanas et al., 2021a), respectively.

Other contributions related to the topics of this thesis are the creation and release of a dataset for sound event detection in urban environments (Zinemanas et al., 2019b), and the collaboration in the development of a dataset for audio-visual scene understanding (Fuentes et al., 2021) and in the development of a Python library for working with environmental sound datasets (Fuentes et al., 2021). The complete list of the author's publications is provided in Appendix B.

7.3 Discussion

This section discusses the advantages and limitations of the type of intrinsic interpretable models proposed in this thesis. Post-hoc methods may be preferred over intrinsic interpretable models because they can explain every black-box model without retraining or redesigning. However, as explained in Section 1.4, post-hoc methods tend to be not very reliable. For instance, the LIME method fits a proxy model, which is a linear approximation of a non-linear model and thus may be an inaccurate representation in some parts of the input space. Besides, confirmation biases can obscure the method evaluation and produce overfitting for some applications (Nickerson, 1998; Dinu et al., 2020; Kaur et al., 2020). For instance, saliency methods were tuned to explain CNN for image classification, so they are not necessarily suitable for audio-based systems. On the contrary, intrinsic interpretable models are designed so that their explanations are faithful to what the model computes and hence more reliable.

For this reason, this thesis focuses on intrinsic interpretable deep-learning models. However, despite the several advantages of intrinsic interpretable models, there are also some limitations. In the following, we discuss some of the main benefits and limitations of the models proposed in this thesis.

7.3.1 Benefits

Since intrinsic interpretable modes are domain-specific, their design process can leverage domain knowledge to improve their discrimination capacity. Therefore we can leverage this knowledge to improve the discriminative capacity of the models. For instance, in APNet (Chapter 5), we propose a similarity measure based on a trainable weighted sum of a frequency-dependent distance in a latent space. Our experiments show that including the trainable weighted sum effectively improves the model, particularly when classifying input data containing mixed sound sources.

Besides, interpretable architectures like APNet allow designers to refine, debug, and improve the model. In this regard, we propose two automatic methods for network refinement that eliminate redundant prototypes and channels. After these refinement processes, we show that the model improves the results, even outperforming the most competitive baseline in one of the tasks.

In light of the above, our results exemplify that intrinsic interpretability may also help to design more accurate models. This contrasts with the widely extended assumption that there is an unavoidable trade-off between interpretability and accuracy.

In addition, inspecting the network allows for evaluating its performance beyond the typical accuracy measure and provides valuable insights into the model's inner workings. Finally, we argue that the interpretability of the proposed models in the form of prototypes (Chapter 5) and attention maps (Chapter 6) increases their trustworthiness. This is important for end-users relying on the network output for actionable decisions, even in low-risk applications.

7.3.2 Limitations and challenges

Since intrinsic interpretable models are domain and application-specific, it is not straightforward to build a model suitable for different applications and domains. For instance, although we evaluate the prototype-based model proposed in Chapter 5 in three different audio contexts (environmental sounds, music, and speech), this model is limited in representing sound mixtures. Moreover, the extension proposed in Chapter 6 to solve this issue is suitable for sound event detection in urban environments but does not necessarily extend to other types of audio signals. For example, given the harmonic structure of music signals, these might not be adequately represented using the local prototypes proposed in the model. Therefore, a further extension is needed to represent such signals.

An exciting research line in the audio community is the development of general representations that can serve for different downstream tasks. Developing interpretable representations of this type is especially interesting, but more research is needed. In Section 7.4, we propose some possible future research directions in this respect.

Another limitation of intrinsic models is that they are usually architecture specific. For instance, the models proposed in this thesis are discriminative CNNs since they are the most used architectures for ESR. However, other relevant systems such as CRNN and Transformers are also used and can produce excellent results. Also, generative and deep reinforcement models are used in the context of audio-domain tasks. However, the interpretable mechanisms proposed in this thesis can not be easily applied to these architectures. Therefore we foresee the development of layers that can bring interpretability to any architecture, such as Concept Whitening (Chen et al., 2020).

7.4 Directions for future research

In this section, we explore potential research directions that are directly related to this thesis and other promising research directions currently under-explored in the audio domain. In particular, we believe that pursuing the following directions would be highly useful for audio researchers and interpretable deep-learning practitioners:

- Design interpretable models beyond the classic CNN-based discriminative architecture.
- Study interpretable audio representations that are useful for different downstream tasks.
- Develop interactive visualizations for human exploration.
- Leverage interpretability to devise human-in-the-loop training strategies.
- Take advantage of interpretability to debug deep-learning models that have learned spurious correlations.
- Build software tools to facilitate the development and use of interpretable methods in the audio domain.

In the rest of this section, we discuss these research directions.

7.4.1 Beyond CNNs and discriminative models

In Section 7.3.2, we mentioned the limitations of the proposed models to extend to other architectures. This limitation widely exists in the interpretability field. Most work in interpretability research has focused on CNN, particularly for image classification. The reason is that the excellent results of CNN trained in large datasets for image classification tasks motivated the deep-learning paradigm shift. Therefore, standard implementations of these architectures were available and became trendy very early (Fong, 2020).

However, several other research areas beyond sound classification or sound event detection could benefit from interpretable models in the audio domain– for instance, sound localization (Politis et al., 2020), music synthesis (Ramires et al., 2020), or sound source separation (Chandna et al., 2017). Besides, DNNs can be trained not to discriminate between classes but to learn high-level representations suitable for different discriminative tasks. The following section shows how interpretability could be applied to these models.

7.4.2 Interpretable deep audio representations

Deep-learning models can extract high-level representations that can lend to different discriminative tasks, e.g., by training a simple MLP on their embeddings (Turian et al., 2022), or even with randomly weighted architectures (Pons & Serra, 2019). Moreover, these representations can be learned using self-supervised learning to avoid manual supervision. For instance, meaningful representations can be learned using contrastive losses over multi-modal correspondence, between audio and tags (Favory et al., 2020; Elizalde et al., 2022), between video and audio (Cramer et al., 2019), or between audio, image, and text (Wu et al., 2022). However, the only constraint imposed on these latent spaces is the correspondence between modalities.

Usually, these models extract representations for each modality and project them to the same space. Ramaswamy et al. (2022) propose to learn low-rank subspaces to gain insights into potential concepts learned within these representations. Furthermore, orthogonalization methods have been used to regularize the latent space (Lezama et al., 2018) and debiasing CNNs (David et al., 2020). Chen et al. (2020) demonstrated that these orthogonal spaces could rotate to align with pre-defined concepts. Thus, it would be interesting to include similar constraints to learn meaningful interpretable deep audio representations.

Moreover, these concept-based spaces can be learned using human feedback to include the users' intuitive sense of the concepts (Lage & Doshi-Velez, 2020). In the next section, we propose other methods to include humans in the training loop using interpretable models.

7.4.3 Human-in-the-loop training strategies

Human-in-the-loop strategies have been mainly focused on feature engineering (Cheng & Bernstein, 2015) and active data labeling (Monarch, 2021). However, interpretable machine learning can enable human interaction during training in order to modify the model. (Lage et al., 2018). For example, Chapter 5 showed how it is possible to design tools to interact with an intrinsic interpretable model (APNet). The user can navigate in a two-dimensional (2D) representation of the prototypes and the training data where he can remove selected prototypes and convert instances to prototypes. Then, once the prototypes are changed, the model can be re-trained and evaluated. These tools allow human-in-the-loop training strategies; for instance, the user can periodically check the training and change the model after a few epochs. We believe this research direction is of interest to the machine learning practitioners working on audio, especially for problems where expert knowledge is fundamental, for instance, from musicologists.

As shown in Chapter 5, interactive visualization tools are useful to implement human-in-the-loop strategies (Sacha et al., 2017). Also, visual explanations obtained in Chapter 6 illustrate the relevance of this kind of tools in interpretability research. The following section explores this area in more depth.

7.4.4 Interactive visualization

Several visualization tools have been proposed to interact with different parts or aspects of a deep neural network, especially CNNs for image classification. For instance, Fong et al. (2021) present an interactive web tool that highlights similarities between patches across different images. Moreover, visualization tools can be devised to illustrate feature inversion methods (Hohman et al., 2020) or explore the network's latent representations and feature interaction (Carter et al., 2019). Finally, several interpretable methods can be treated as composable building blocks to produce enriched interfaces (Olah et al., 2018). We believe that similar tools should be developed for explaining and interacting with models in the audio domain.

Although these tools are of great value to the community, more research is needed to develop meaningful tools for human interaction with models, particularly during training. In addition, the methods presented above are for post-hoc explanations. So, specific visualization tools to interact with intrinsic models are needed. We consider the visualization tool for APNet as an attempt in that direction, but more research and development are needed.

7.4.5 Model debugging and refining

Interpretable methods can help to refine and debug a deep-learning model. For instance, layer-wise relevance methods have been used to detect spurious correlations in the data (Anders et al., 2019). However, Adebayo et al. (2022) argue that post-hoc explanations can be used to identify spurious correlations only for known and visible spurious signals. Moreover, Sun et al. (2020) proposed a layer-wise inference fine-tuning strategy to improve an image captioning model. Interpretable methods can also help to detect and mitigate biases reproduced or generated by the models (Kim et al., 2018; David et al., 2020).

With regards to model refining, Section 5.4.2 illustrated how intrinsically interpretable layers can help to refine the model either automatically or manually, reducing the number of parameters and even improving the performance. In the case of post-hoc explainability, Yeom et al. (2021) propose a network pruning method to find the model's most relevant units. Besides, information theory can be use to prune the networks (Davis et al., 2020). Unfortunately, to the best of our knowledge, using interpretability to refine, prune or find spurious correlation has not been studied yet in the audio domain, apart from the preliminary experiments presented in Chapter 5. For this reason, we foresee this kind of work as a promising direction for future research.

7.4.6 Tools for audio interpretability

The development of tools for audio interpretability is another relevant direction for future research. Interpretable post-hoc methods are usually hard to implement, and some available implementations are not standardized. Besides, most of the existing methods and their implementations are devised for image classification networks. However, audio-related tasks have several specifities that should be taken into account. For instance, pre-processing steps, such as, re-sampling, scaling, data augmentation, and feature extraction, are typically needed before inference. In particular, feature extraction typically involves the calculation of a time-frequency representation which are very sensitive to the parameters and implementation details. This calls for specific software tools that help simplify the development of interpretable deep-learning models in the audio domain.

For instance, the iNNvestigate library provides a standard interface and out-of-the-box implementation for many post-hoc explainable methods (Alber et al., 2019). It would be interesting to develop a similar library for audio the domain, including methods for sound models such as SLIME. Developing such a tool would simplify the systematic comparison between interpretable methods.

Regarding intrinsically interpretable methods, building a joint implementation for several methods is more challenging because each model has its characteristics and functionalities. However, using a common development framework such as DCASE-models (Chapter 3) would contribute to the standardization of the implementation of such models.

Pablo Zinemanas Friet, Barcelona, 20 July 2023.

Bibliography

Aamodt, A. & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7, 39–59. `https://dl.acm.org/doi/10.5555/196108.196115`. (Cited on page 32.)

Abeßer, J. (2021). USM-SED - A Dataset for Polyphonic Sound Event Detection in Urban Sound Monitoring Scenarios. *CoRR*, abs/2105.02592. `https://arxiv.org/abs/2105.02592`. (Cited on page 27.)

Abeßer, J., Gourishetti, S., Kátai, A., Clauß, T., Sharma, P., & Liebetrau, J. (2021). IDMT-Traffic: An Open Benchmark Dataset for Acoustic Traffic Monitoring Research. In *29th European Signal Processing Conference, EUSIPCO 2021* (pp. 551–555).: IEEE. `https://doi.org/10.23919/EUSIPCO54536.2021.9616080`. (Cited on page 27.)

Abrol, V., Sharma, P., & Thakur, A. (2017). *GMM-AA System for Acoustic Scene Classification.* Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 20.)

Abushariah, M. & Sawalha, M. (2013). The effects of speakers' gender, age, and region on overall performance of Arabic automatic speech recognition systems using the phonetically rich and balanced Modern Standard Arabic speech corpus. In *Proceedings of the 2nd Workshop of Arabic Corpus Linguistics WACL-2* Lancaster, UK. (Cited on page 4.)

Adavanne, S., Parascandolo, G., Pertila, P., Heittola, T., & Virtanen, T. (2016). Sound Event Detection in Multichannel Audio Using Spatial and Harmonic Features. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)* (pp. 6–10). (Cited on pages xvii, 17, 24, 25, and 27.)

Adavanne, S., Pertilä, P., & Virtanen, T. (2017). Sound event detection using spatial features and convolutional recurrent neural network. In *2017 International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 771–775). New Orleans, LA, USA: IEEE. `https://doi.org/10.1109/ICASSP.2017.7952260`. (Cited on pages 46 and 47.)

Adavanne, S. & Virtanen, T. (2017). *A Report on Sound Event Detection with Different Binaural Features.* Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 17.)

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity Checks for Saliency Maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 31: Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf`. (Cited on page 6.)

Adebayo, J., Muelly, M., Abelson, H., & Kim, B. (2022). Post hoc Explanations may be Ineffective for Detecting Unknown Spurious Correlation. In *International Conference on Learning Representations*. `https://openreview.net/forum?id=xNOVfCCvDpM`. (Cited on page 103.)

Afchar, D., Hennequin, R., & Guigue, V. (2022). Learning Unsupervised Hierarchies of Audio Concepts. *CoRR*, abs/2207.11231. `https://doi.org/10.48550/arXiv.2207.11231`. (Cited on page 35.)

Agrawal, P. & Ganapathy, S. (2020). Interpretable Representation Learning for Speech and Audio Signals Based on Relevance Weighting. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 2823–2836. `https://doi.org/10.1109/TASLP.2020.3030489`. (Cited on pages 35 and 36.)

Alber, M., et al. (2019). iNNvestigate Neural Networks! *Journal of Machine Learning Research*, 20(93), 1–8. `http://jmlr.org/papers/v20/18-540.html`. (Cited on page 104.)

Alvarez Melis, D. & Jaakkola, T. (2018). Towards Robust Interpretability with Self-Explaining Neural Networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 31 (pp. 7775–7784).: Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2018/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf`. (Cited on page 7.)

Anders, C. J., Marinc, T., Neumann, D., Samek, W., Müller, K., & Lapuschkin, S. (2019). Analyzing ImageNet with Spectral Relevance Analysis: Towards ImageNet un-Hans'ed. *CoRR*, abs/1912.11425. `http://arxiv.org/abs/1912.11425`. (Cited on page 103.)

Andrews, R., Diederich, J., & Tickle, A. B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6), 373–389. Knowledge-based neural networks. `https://doi.org/10.1016/0950-7051(96)81920-4`. (Cited on page 30.)

Ardizzone, L., Kruse, J., Rother, C., & Köthe, U. (2019). Analyzing Inverse Problems with Invertible Neural Networks. In *International Conference on*

Learning Representations New Orleans, LA, USA. https://openreview.net/forum?id=rJed6j0cKX. (Cited on page 35.)

Arik, S. Ö. & Pfister, T. (2020). ProtoAttend: Attention-Based Prototypical Learning. *Journal of Machine Learning Research*, 21, 210:1–210:35. http://jmlr.org/papers/v21/20-042.html. (Cited on page 33.)

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS ONE*, 7(10). https://doi.org/10.1371/journal.pone.0130140. (Cited on page 29.)

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615. (Cited on page 69.)

Barredo Arrieta, A., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012. (Cited on pages 9, 28, 30, and 39.)

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network Dissection: Quantifying Interpretability of Deep Visual Representations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (pp. 3319–3327).: IEEE Computer Society. https://doi.org/10.1109/CVPR.2017.354. (Cited on page 30.)

Becker, S., Ackermann, M., Lapuschkin, S., Müller, K., & Samek, W. (2018). Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals. *CoRR*, abs/1807.03418. http://arxiv.org/abs/1807.03418. (Cited on page 34.)

Bello, J. P., Mydlarz, C., & Salamon, J. (2018). *Sound Analysis in Smart Cities*, (pp. 373–397). Springer International Publishing: Cham. https://doi.org/10.1007/978-3-319-63450-0_13. (Cited on pages 2 and 3.)

Benetos, E., Lafay, G., Lagrange, M., & Plumbley, M. D. (2016). Detection of overlapping acoustic events using a temporally-constrained probabilistic model. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6450–6454). Shanghai, China: IEEE. https://doi.org/10.1109/ICASSP.2016.7472919. (Cited on page 20.)

Berry, M. W., Browne, M., Langville, A. N., Pauca, V. P., & Plemmons, R. J. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1), 155–173. https://doi.org/10.1016/j.csda.2006.11.006. (Cited on page 32.)

Bichindaritz, I. & Marling, C. (2006). Case-based reasoning in the health sciences: What's next? *Artificial Intelligence in Medicine*, 36(2), 127–135. Case-based reasoning in the health sciences. https://doi.org/10.1016/j.artmed.2005.10.008. (Cited on page 32.)

Bien, J. & Tibshirani, R. (2011). Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4). https://doi.org/10.1214/11-aoas495. (Cited on page 33.)

Bilen, C., Ferroni, G., Tuveri, F., Azcarreta, J., & Krstulovic, S. (2020). A Framework for the Robust Evaluation of Sound Event Detection. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 61–65). Barcelona, Spain: IEEE. https://doi.org/10.1109/ICASSP40776.2020.9052995. (Cited on pages 9 and 47.)

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. (Cited on page 22.)

Bisot, V., Serizel, R., Essid, S., & Richard, G. (2016). *Supervised Nonnegative Matrix Factorization for Acoustic Scene Classification*. Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 20.)

Bisot, V., Serizel, R., Essid, S., & Richard, G. (2017). *Nonnegative Feature Learning Methods for Acoustic Scene Classification*. Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 22.)

Bittner, R., Salamon, J., Tierney, M., Mauch, M., Cannam, C., & Bello, J. (2014). MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference* Taipei, Taiwan: ISMIR. https://doi.org/10.5281/zenodo.1417889. (Cited on page 73.)

Bittner, R. M., Humphrey, E. J., & Bello, J. P. (2016). pysox: Leveraging the Audio Signal Processing Power of SoX in Python. In *Proceedings of the 17th International Society for Music Information Retrieval Conference* New York City, USA. (Cited on page 43.)

Bjornson, E. (2019). Reproducible Research: Best Practices and Potential Misuse [Perspectives]. *IEEE Signal Processing Magazine*, 36(3), 106–123. https://doi.org/10.1109/MSP.2019.2898421. (Cited on pages 9 and 39.)

Brown, J. C. (1991). Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, 89, 425–434. (Cited on page 19.)

Cakir, E., Parascandolo, G., Heittola, T., Huttunen, H., & Virtanen, T. (2017). Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. *Transactions on Audio, Speech and Language Processing: Special issue on Sound Scene and Event Analysis*, 25(6), 1291–1303. `https://dl.acm.org/doi/10.1109/TASLP.2017.2690575`. (Cited on pages xvii, 17, 18, 25, 27, and 34.)

Cakır, E. & Virtanen, T. (2018). End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input. In *2018 International Joint Conference on Neural Networks (IJCNN)* Rio de Janeiro, Brazil: IEEE. `https://doi.org/10.1109/IJCNN.2018.8489470`. (Cited on pages 37, 54, and 61.)

Carlini, N. & Wagner, D. (2018). Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *IEEE Security and Privacy Workshops (SPW)* (pp. 1–7). San Francisco, CA, USA: IEEE. `https://doi.org/10.1109/SPW.2018.00009`. (Cited on page 1.)

Carter, S., Armstrong, Z., Schubert, L., Johnson, I., & Olah, C. (2019). Activation Atlas. *Distill*. `https://doi.org/10.23915/distill.00015`. (Cited on page 103.)

Cartwright, M., et al. (2020). SONYC-UST-V2: An Urban Sound Tagging Dataset with Spatiotemporal Context. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)* (pp. 16–20). Tokyo, Japan. (Cited on page 26.)

Cartwright, M., et al. (2019). SONYC Urban Sound Tagging (SONYC-UST): A Multilabel Dataset from an Urban Acoustic Sensor Network. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)* (pp. 35–39). New York, USA. (Cited on pages 9 and 41.)

Chaki, S., Doshi, P., Bhattacharya, S., & Patnaik, P. (2020). Explaining perceived emotion predictions in music: An attentive approach. In *Proceedings of the 21st International Society for Music Information Retrieval Conference* (pp. 150–156). Montréal, Canada: ISMIR. `https://doi.org/10.5281/zenodo.4245388`. (Cited on page 36.)

Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4960–4964). Shanghai, China: IEEE. `https://doi.org/10.1109/ICASSP.2016.7472621`. (Cited on page 35.)

Chandna, P., Miron, M., Janer, J., & Gómez, E. (2017). Monoaural Audio Source Separation Using Deep Convolutional Neural Networks. In

P. Tichavský, M. Babaie-Zadeh, O. J. Michel, & N. Thirion-Moreau (Eds.), *Latent Variable Analysis and Signal Separation* (pp. 258–266). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-53547-0_25. (Cited on page 101.)

Chen, C., Li, O., Tao, D., Barnett, A. J., Su, J., & Rudin, C. (2019a). This Looks Like That: Deep Learning for Interpretable Image Recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 32: Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf. (Cited on pages 5, 7, 33, 87, and 95.)

Chen, L., et al. (2019b). Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening. *PLOS ONE*, 14(8), 1–22. https://doi.org/10.1371/journal.pone.0220113. (Cited on page 1.)

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16 (pp. 2180–2188). Red Hook, NY, USA: Curran Associates Inc. https://proceedings.neurips.cc/paper/2016/hash/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Abstract.html. (Cited on page 32.)

Chen, Z., Bei, Y., & Rudin, C. (2020). Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12), 772–782. https://doi.org/10.1038/s42256-020-00265-z. (Cited on pages 7, 28, 31, 100, and 102.)

Cheng, J. & Bernstein, M. S. (2015). Flock: Hybrid Crowd-Machine Learning Classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15 (pp. 600–611). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2675133.2675214. (Cited on page 102.)

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Doha, Qatar. https://doi.org/10.3115/v1/d14-1179. (Cited on page 24.)

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Transfer learning for music classification and regression tasks. In *Proceedings of the 18th Interna-*

tional Society for Music Information Retrieval Conference Suzhou, China: ISMIR. https://doi.org/10.5281/zenodo.1418015. (Cited on page 35.)

Chollet, F. et al. (2015). Keras. https://keras.io. (Cited on page 59.)

Chorowski, J., Bahdanau, D., Cho, K., & Bengio, Y. (2014). End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results. *CoRR*, abs/1412.1602. http://arxiv.org/abs/1412.1602. (Cited on page 35.)

Chowdhury, S., Vall, A., Haunschmid, V., & Widmer, G. (2019). Towards Explainable Music Emotion Recognition: The Route via Mid-level Features. In *Proceedings of the 20th International Society for Music Information Retrieval Conference* (pp. 237–243). Delft, The Netherlands: ISMIR. https://doi.org/10.5281/zenodo.3527788. (Cited on page 35.)

Chu, S., Narayanan, S., & Kuo, C.-C. J. (2009). Environmental Sound Recognition With Time–Frequency Audio Features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6), 1142–1158. https://doi.org/10.1109/TASL.2009.2017438. (Cited on page 20.)

Colom, M., Kerautret, B., Limare, N., Monasse, P., & Morel, J. (2015). IPOL: A new journal for fully reproducible research; analysis of four years development. In *Proceedings of the 7th International Conference on New Technologies, Mobility and Security (NTMS)* (pp. 1–5). https://doi.org/10.1109/NTMS.2015.7266500. (Cited on pages 9 and 39.)

Cramer, J., Lostanlen, V., Farnsworth, A., Salamon, J., & Bello, J. P. (2020). Chirping up the Right Tree: Incorporating Biological Taxonomies into Deep Bioacoustic Classifiers. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 901–905). Barcelona, Spain: IEEE. https://doi.org/10.1109/ICASSP40776.2020.9052908. (Cited on pages 2 and 27.)

Cramer, J., Wu, H.-H., Salamon, J., & Bello, J. P. (2019). Look, Listen and Learn More: Design Choices for Deep Audio Embeddings. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3852–3856). Brighton, United Kingdom: IEEE. https://doi.org/10.1109/ICASSP.2019.8682475. (Cited on pages 44, 74, 91, and 102.)

Crocco, M., Cristani, M., Trucco, A., & Murino, V. (2016). Audio Surveillance: A Systematic Review. *ACM Computing Surveys*, 48(4). https://doi.org/10.1145/2871183. (Cited on page 2.)

Daniel Steele, D. K. & Guastavino, C. (2013). The Sensor City Initiative: cognitive sensors for soundscape transformations. In *Geoinformatics for City*

Transformations (pp. 243–253).: Technical University of Ostrava. (Cited on page 2.)

David, K. E., Liu, Q., & Fong, R. (2020). Debiasing Convolutional Neural Networks via Meta Orthogonalization. *CoRR*, abs/2011.07453. `https://arxiv.org/abs/2011.07453`. (Cited on pages 4, 102, and 103.)

Davis, B., Bhatt, U., Bhardwaj, K., Marculescu, R., & Moura, J. M. (2020). On Network Science and Mutual Information for Explaining Deep Neural Networks. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8399–8403). Barcelona, Spain: IEEE. `https://doi.org/10.1109/ICASSP40776.2020.9053078`. (Cited on page 103.)

Davis, S. & Mermelstein, P. (1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 357–366. (Cited on page 18.)

de Andrade, D. C., Leo, S., Viana, M. L. D. S., & Bernkopf, C. (2018). A neural attention model for speech command recognition. *CoRR*, abs/1808.08929. `http://arxiv.org/abs/1808.08929`. (Cited on page 74.)

de Berardinis, J., Cangelosi, A., & Coutinho, E. (2020). The multiple voices of musical emotions: source separation for improving music emotion recognition models and their interpretability. In *Proceedings of the 21st International Society for Music Information Retrieval Conference* Montréal, Canada: ISMIR. `https://doi.org/10.5281/zenodo.4245428`. (Cited on page 35.)

Dieleman, S. & Schrauwen, B. (2014). End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6964–6968). Florence, Italy: IEEE. `https://doi.org/10.1109/ICASSP.2014.6854950`. (Cited on page 53.)

Dinu, J., Bigham, J. P., & Kolter, J. Z. (2020). Challenging common interpretability assumptions in feature attribution explanations. *CoRR*, abs/2012.02748. `https://arxiv.org/abs/2012.02748`. (Cited on page 99.)

Dorfer, M., Lehner, B., Eghbal-zadeh, H., Christop, H., Fabian, P., & Gerhard, W. (2018). *Acoustic Scene Classification with Fully Convolutional Neural Networks and I-Vectors*. Technical report, Detection and Classification of Acoustic Scenes and Events 2018 Challenge. (Cited on page 24.)

Dotan, R. & Milli, S. (2020). Value-Laden Disciplinary Shifts in Machine Learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20 (pp. 294). New York, NY, USA: Association for Computing Machinery. `https://doi.org/10.1145/3351095.3373157`. (Cited on pages 3 and 4.)

Drossos, K., Lipping, S., & Virtanen, T. (2020). Clotho: an Audio Captioning Dataset. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 736–740). Barcelona, Spain: IEEE. https://doi.org/10.1109/ICASSP40776.2020.9052990. (Cited on page 27.)

Drugman, T., Urbain, J., Bauwens, N., Chessini, R., Aubriot, A.-S., Lebecque, P., & Dutoit, T. (2012). Audio and Contact Microphones for Cough Detection. In *Proceedings of INTERSPEECH 2012* (pp. 1303–1306). Portland, OR, USA: ISCA. http://www.isca-speech.org/archive/interspeech_2012/i12_1303.html. (Cited on page 2.)

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12, 2121–2159. http://dl.acm.org/citation.cfm?id=1953048.2021068. (Cited on page 22.)

Ebbers, J. & Häb-Umbach, R. (2019). Convolutional Recurrent Neural Network and Data Augmentation for Audio Tagging with Noisy Labels and Minimal Supervision. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)* (pp. 64–68). New York University, NY, USA. (Cited on page 25.)

Elizalde, B., Deshmukh, S., Ismail, M. A., & Wang, H. (2022). CLAP: Learning Audio Concepts From Natural Language Supervision. *CoRR*, abs/2206.04769. https://doi.org/10.48550/arXiv.2206.04769. (Cited on page 102.)

Elizalde, B., Kumar, A., Shah, A., Badlani, R., Vincent, E., Raj, B., & Lane, I. (2016). Experiments on the DCASE Challenge 2016: Acoustic Scene Classification and Sound Event Detection in Real Life Recording. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)* (pp. 20–24). (Cited on page 18.)

Favory, X., Drossos, K., Virtanen, T., & Serra, X. (2020). COALA: Co-Aligned Autoencoders for Learning Semantically Enriched Audio Representations. In *ICML 2020 Workshop on Self-supervision in Audio and Speech*. https://openreview.net/forum?id=7jxwhNDM0Uv. (Cited on page 102.)

Fletcher, H. (1940). Auditory Patterns. *Reviews of Modern Physics*, 12, 47–65. https://doi.org/10.1103/RevModPhys.12.47. (Cited on page 19.)

Foleiss, J. & Tavares, T. (2017). *MLP-Based Feature Learning for Automatic Acoustic Scene Classification*. Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 22.)

Fong, R. (2020). *Understanding Convolutional Neural Networks*. PhD thesis, St. John's College, University of Oxford. (Cited on page 101.)

Fong, R., Mordvintsev, A., Vedaldi, A., & Olah, C. (2021). Interactive Similarity Overlays. In *VISxAI*. `https://www.ruthfong.com/projects/interactive_overlay/`. (Cited on page 103.)

Fonseca, E., Favory, X., Pons, J., Font, F., & Serra, X. (2022). FSD50K: An Open Dataset of Human-Labeled Sound Events. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 30, 829–852. `https://doi.org/10.1109/TASLP.2021.3133208`. (Cited on page 26.)

Fonseca, E., Plakal, M., Font, F., Ellis, D. P. W., Favory, X., Pons, J., & Serra, X. (2018). General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)* Surrey, UK. (Cited on pages 9 and 41.)

Font, F., Roma, G., & Serra, X. (2013). Freesound Technical Demo. In *ACM International Conference on Multimedia (MM'13)* (pp. 411–412). Barcelona, Spain: ACM ACM. `https://doi.org/10.1145/2502081.2502245`. (Cited on page 73.)

Foscarin, F., Hoedt, K., Praher, V., Flexer, A., & Widmer, G. (2022). Concept-Based Techniques for "Musicologist-friendly" Explanations in a Deep Music Classifier. *CoRR*, abs/2208.12485. `https://doi.org/10.48550/arXiv.2208.12485`. (Cited on page 35.)

Fuentes, M., et al. (2021). Soundata: A Python library for reproducible use of audio datasets. *CoRR*, abs/2109.12690. `https://arxiv.org/abs/2109.12690`. (Cited on pages 3, 11, and 99.)

Fuentes, M., et al. (2022). Urban Sound & Sight: Dataset And Benchmark For Audio-Visual Urban Scene Understanding. In *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 141–145). Singapore: IEEE. `https://doi.org/10.1109/ICASSP43922.2022.9747644`. (Cited on pages 3 and 11.)

Garnerin, M., Rossato, S., & Besacier, L. (2019). Gender Representation in French Broadcast Corpora and Its Impact on ASR Performance. In *Proceedings of the 1st International Workshop on AI for Smart TV Content Production, Access and Delivery*. (Cited on page 4.)

Gemmeke, J. F., et al. (2017). Audio Set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 776–780). New Orleans, LA, USA: IEEE. `https://doi.org/10.1109/ICASSP.2017.7952261`. (Cited on page 26.)

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining Explanations: An Overview of Interpretability of Machine Learning. In *5th International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 80–89). Turin, Italy: IEEE. `https://doi.org/10.1109/DSAA.2018.00018`. (Cited on pages xvii, 2, 5, 6, 28, 29, 30, 31, and 32.)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. (Cited on pages 1, 21, 23, and 24.)

Goodman, B. & Flaxman, S. (2017). European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Magazine*, 38(3), 50–57. `https://doi.org/10.1609/aimag.v38i3.2741`. (Cited on page 1.)

Gorin, A., Makhazhanov, N., & Shmyrev, N. (2016). *DCASE 2016 Sound Event Detection System based on Convolutional Neural Network*. Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 17.)

Griffin, D. & Lim, J. (1984). Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2), 236–243. (Cited on page 77.)

Guidotti, R., Monreale, A., Turini, F., Pedreschi, D., & Giannotti, F. (2018). A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51. `https://doi.org/10.1145/3236009`. (Cited on page 5.)

Harb, R. & Pernkopf, F. (2018). Sound event detection using weakly labelled semi-supervised data with GCRNNs, VAT and self-adaptive label refinement. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)* (pp. 83–87). (Cited on page 25.)

Hase, P., Chen, C., Li, O., & Rudin, C. (2019). Interpretable Image Recognition with Hierarchical Prototypes. In *The Seventh AAAI Conference on Human Computation and Crowdsourcing (HCOMP-19)*, volume 7 (pp. 32–40). Stevenson, WA, USA: AAAI Press. (Cited on pages 7 and 33.)

Haunschmid, V., Manilow, E., & Widmer, G. (2020). audioLIME: Listenable Explanations Using Source Separation. *CoRR*, abs/2008.00582. `https://arxiv.org/abs/2008.00582`. (Cited on page 34.)

Heittola, T., Mesaros, A., & Virtanen, T. (2020). Acoustic scene classification in DCASE 2020 Challenge: generalization across devices and low complexity solutions. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)* (pp. 56–60). Tokyo, Japan. (Cited on pages 9 and 41.)

Herrera-Boyer, P., Klapuri, A., & Davy, M. (2006). *Automatic Classification of Pitched Musical Instrument Sounds*, (pp. 163–200). Springer, New York. (Cited on page 18.)

Hershey, S., et al. (2017). CNN Architectures for Large-Scale Audio Classification. In *2017 International Conference on Acoustics, Speech and Signal Processing (ICASSP)* New Orleans, LA, USA: IEEE. `https://arxiv.org/abs/1609.09430`. (Cited on pages 40, 44, and 46.)

Higgins, I., et al. (2017). beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*. `https://openreview.net/forum?id=Sy2fzU9gl`. (Cited on page 32.)

High Level Expert Group on Artificial Intelligence (2019). *Ethics guidelines for trustworthy AI*. Technical report, European Commission. (Cited on pages 9 and 39.)

Hinton, G., et al. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6), 82–97. `https://doi.org/10.1109/MSP.2012.2205597`. (Cited on page 1.)

Hochreiter, S. & Schmidhuber, J. (1997). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Neural Computation*, 9, 1735–1780. (Cited on page 24.)

Hohman, F., Park, H., Robinson, C., & Chau, D. H. P. (2020). Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations. *IEEE Transactions on Visualization and Computer Graphics*, 26, 1096–1106. `https://doi.org/10.1109/TVCG.2019.2934659`. (Cited on page 103.)

Huang, C. A., et al. (2019). Music Transformer: Generating Music with Long-Term Structure. In *7th International Conference on Learning Representations, ICLR 2019*: OpenReview.net. `https://openreview.net/forum?id=rJe4ShAcF7`. (Cited on page 36.)

Hyvärinen, A. & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4), 411–430. `https://doi.org/10.1016/S0893-6080(00)00026-5`. (Cited on page 32.)

Hüwel, A., Adiloğlu, K., & Bach, J.-H. (2020). Hearing aid Research Data Set for Acoustic Environment Recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 706–710). Barcelona, Spain: IEEE. `https://doi.org/10.1109/ICASSP40776.2020.9053611`. (Cited on page 2.)

Jalal, M. A., Milner, R., & Hain, T. (2020). Empirical Interpretation of Speech Emotion Perception with Attention Based Model for Speech Emotion Recognition. In *Proceedings of INTERSPEECH 2020* (pp. 4113–4117). Shanghai, China: ISCA. `http://doi.org/10.21437/Interspeech.2020-3007`. (Cited on page 36.)

Jeong, I.-Y., Lee, S., Han, Y., & Lee, K. (2017). *Audio Event Detection Using Multiple-Input Convolutional Neural Network*. Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 17.)

Joder, C., Essid, S., & Richard, G. (2009). Temporal Integration for Audio Classification With Application to Musical Instrument Classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(1), 174–186. `https://doi.org/10.1109/TASL.2008.2007613`. (Cited on page 73.)

Jolliffe, I. T. (1986). *Principal Component Analysis and Factor Analysis*, (pp. 115–128). Springer New York: New York, NY. `https://doi.org/10.1007/978-1-4757-1904-8_7`. (Cited on page 32.)

Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H., & Wortman Vaughan, J. (2020). Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20 (pp. 1–14). New York, NY, USA: Association for Computing Machinery. `https://doi.org/10.1145/3313831.3376219`. (Cited on page 99.)

Kelz, R. & Widmer, G. (2019). Towards Interpretable Polyphonic Transcription with Invertible Neural Networks. In *Proceedings of the 20th International Society for Music Information Retrieval Conference* (pp. 376–383). Delft, The Netherlands: ISMIR. `https://doi.org/10.5281/zenodo.3527822`. (Cited on page 35.)

Kim, B., Rudin, C., & Shah, J. A. (2014). The Bayesian Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, volume 27 (pp. 1952–1960). Montreal, Quebec, Canada. `https://proceedings.neurips.cc/paper/2014/hash/390e982518a50e280d8e2b535462ec1f-Abstract.html`. (Cited on pages 32 and 33.)

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., & sayres, R. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research* (pp. 2668–2677). Stockholmsmässan, Stockholm Sweden: PMLR. http://proceedings.mlr.press/v80/kim18d.html. (Cited on pages 6, 28, 31, and 103.)

Kim, Y. E. & Whitman, B. (2002). Singer identification in popular music recordings using voice coding features. In *Proceedings of the 3rd International Conference on Music Information Retrieval* (pp. 164–169). (Cited on page 18.)

Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980. http://arxiv.org/abs/1412.6980. (Cited on page 22.)

Koenecke, A., et al. (2020). Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14), 7684–7689. https://doi.org/10.1073/pnas.1915768117. (Cited on page 4.)

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., & Liang, P. (2020). Concept Bottleneck Models. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research* (pp. 5338–5348).: PMLR. https://proceedings.mlr.press/v119/koh20a.html. (Cited on page 31.)

Kong, Q., Sobieraj, I., Wang, W., & Plumbley, M. (2016). Deep Neural Network Baseline for DCASE Challenge 2016. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)* (pp. 50–54). (Cited on page 18.)

Krishnan, M. (2020). Against Interpretability: a Critical Examination of the Interpretability Problem in Machine Learning. *Philosophy & Technology*, 33(3), 487–502. https://doi.org/10.1007/s11263-015-0816-y. (Cited on pages 1 and 5.)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105).: Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. (Cited on pages 1 and 53.)

Kuhn, T. S. (1970). *The Structure of Scientific Revolutions*. University of Chicago Press. (Cited on page 3.)

Kun, Q., Zhao, R., Vedhas, P., Zijiang, Y., Zixing, Z., & Björn, S. (2017). *Wavelets Revisited for the Classification of Acoustic Scenes*. Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 20.)

Lage, I. & Doshi-Velez, F. (2020). Learning Interpretable Concept-Based Models with Human Feedback. *International Conference on Machine Learning: Workshop on Human Interpretability in Machine Learning*, 1, 1–11. `https://arxiv.org/abs/2012.02898`. (Cited on page 102.)

Lage, I., Ross, A. S., Kim, B., Gershman, S. J., & Doshi-Velez, F. (2018). Human-in-the-Loop Interpretability Prior. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18 (pp. 10180–10189). Red Hook, NY, USA: Curran Associates Inc. `https://proceedings.neurips.cc/paper/2018/hash/0a7d83f084ec258aefd128569dda03d7-Abstract.html`. (Cited on page 102.)

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 56–67. (Cited on page 1.)

Lee, J., Park, J., Kim, K. L., & Nam, J. (2018). SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification. *Applied Sciences*, 8(1). `https://doi.org/10.3390/app8010150`. (Cited on pages 8, 36, and 53.)

Lee, M. K., Jain, A., Cha, H. J., Ojha, S., & Kusbit, D. (2019). Procedural Justice in Algorithmic Fairness: Leveraging Transparency and Outcome Control for Fair Algorithmic Mediation. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW). `https://doi.org/10.1145/3359284`. (Cited on page 4.)

Leonelli, S. & Ankeny, R. A. (2015). Repertoires: How to Transform a Project into a Research Community. *BioScience*, 65(7), 701–708. (Cited on page 9.)

Lezama, J., Qiu, Q., Musé, P., & Sapiro, G. (2018). OLE: Orthogonal Low-rank Embedding, A Plug and Play Geometric Loss for Deep Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 8109–8118). `https://doi.org/10.1109/CVPR.2018.00846`. (Cited on page 102.)

Li, C., Yuan, P., & Lee, H. (2020). What Does a Network Layer Hear? Analyzing Hidden Representations of End-to-End ASR Through Speech Synthesis. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6434–6438). Barcelona, Spain: IEEE. `https://doi.org/10.1109/ICASSP40776.2020.9054675`. (Cited on page 35.)

Li, H. & Sun, J. (2008). Ranking-order case-based reasoning for financial distress prediction. *Knowledge-Based Systems*, 21(8), 868–878. https://doi.org/10.1016/j.knosys.2008.03.047. (Cited on page 32.)

Li, O., Liu, H., Chen, C., & Rudin, C. (2018a). Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, volume 32 (pp. 3530–3537). New Orleans, Louisiana, USA: AAAI Press. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17082. (Cited on pages 5, 7, 33, 65, 66, 68, 70, 71, 85, and 95.)

Li, Z., Zhang, L., Du, S., & Liu, W. (2018b). *Acoustic Scene Classification Based on Binaural Deep Scattering Spectra with CNN and LSTM*. Technical report, Detection and Classification of Acoustic Scenes and Events 2018 Challenge. (Cited on page 24.)

Lim, H., Park, J., & Han, Y. (2017). Rare Sound Event Detection Using 1D Convolutional Recurrent Neural Networks. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)* (pp. 80–84). (Cited on page 25.)

Lim, W., Suh, S., & Jeong, Y. (2018). Weakly labeled semi-supervised sound event detection using CRNN with inception module. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)* (pp. 74–77). (Cited on page 25.)

Lipton, Z. (2016). The Mythos of Model Interpretability. *Communications of the ACM*, 61. https://doi.org/10.1145/3233231. (Cited on page 5.)

Loiseau, R., Bouvier, B., Teytaut, Y., Vincent, E., Aubry, M., & Landrieu, L. (2022). A Model You Can Hear: Audio Identification with Playable Prototypes. *CoRR*, abs/2208.03311. https://doi.org/10.48550/arXiv.2208.03311. (Cited on page 36.)

Lostanlen, V. & Cella, C.-E. (2016). Deep convolutional networks on the pitch spiral for musical instrument recognition. In *Proceedings of the 17th International Society for Music Information Retrieval Conference* New York, USA: ISMIR. https://doi.org/10.5281/zenodo.1416928. (Cited on page 73.)

Lostanlen, V., Salamon, J., Cartwright, M., McFee, B., Farnsworth, A., Kelling, S., & Bello, J. P. (2019). Per-Channel Energy Normalization: Why and How. *IEEE Signal Processing Letters*, 26(1), 39–43. https://doi.org/10.1109/LSP.2018.2878620. (Cited on pages 18, 19, and 61.)

Loweimi, E., Bell, P., & Renals, S. (2019). On Learning Interpretable CNNs with Parametric Modulated Kernel-Based Filters. In *Proceedings of INTERSPEECH 2019* (pp. 3480–3484). Graz, Austria: ISCA. http://doi.org/10.21437/Interspeech.2019-1257. (Cited on pages 8, 36, and 53.)

Lu, R. & Duan, Z. (2017). *Bidirectional GRU for Sound Event Detection*. Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 18.)

Luo, Y.-J., Cheuk, K. W., Nakano, T., Goto, M., & Herremans, D. (2020). Unsupervised Disentanglement of Pitch and Timbre for Isolated Musical Instrument Sounds. In *Proceedings of the 21st International Society for Music Information Retrieval Conference* Montréal, Canada: ISMIR. `https://doi.org/10.5281/zenodo.4245532`. (Cited on page 35.)

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 Atlanta, Georgia, USA: JMLR. (Cited on page 69.)

Mahendran, A. & Vedaldi, A. (2015). Understanding deep image representations by inverting them. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 5188–5196). `https://doi.org/10.1109/CVPR.2015.7299155`. (Cited on page 30.)

McFee, B. (2018). *Statistical Methods for Scene and Event Classification*, (pp. 103–146). Springer International Publishing: Cham. `https://doi.org/10.1007/978-3-319-63450-0_5`. (Cited on pages 20, 21, 22, 23, and 24.)

McFee, B., Kim, J. W., Cartwright, M., Salamon, J., Bittner, R. M., & Bello, J. P. (2019). Open-Source Practices for Music Signal Processing Research: Recommendations for Transparent, Sustainable, and Reproducible Audio Research. *IEEE Signal Processing Magazine*, 36(1), 128–137. (Cited on pages 8, 9, 39, and 41.)

McFee, B., et al. (2015). librosa: 0.4.1. `https://doi.org/10.5281/zenodo.32193`. (Cited on pages 56 and 77.)

Menon, S., Damian, A., Hu, M., Ravi, N., & Rudin, C. (2020). PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2434–2442). Seattle, WA, USA: IEEE. `https://doi.org/10.1109/CVPR42600.2020.00251`. (Cited on page 1.)

Mesaros, A., Diment, A., Elizalde, B., Heittola, T., Vincent, E., Raj, B., & Virtanen, T. (2019). Sound event detection in the DCASE 2017 Challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(6), 992–1006. `https://doi.org/10.1109/TASLP.2019.2907016`. (Cited on pages 9 and 26.)

Mesaros, A., Heittola, T., Benetos, E., Foster, P., Lagrange, M., Virtanen, T., & Plumbley, M. D. (2018a). Detection and Classification of Acoustic

Scenes and Events: Outcome of the DCASE 2016 Challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(2), 379–393. `https://doi.org/10.1109/TASLP.2017.2778423`. (Cited on pages 3, 9, and 26.)

Mesaros, A., et al. (2017). DCASE 2017 Challenge Setup: Tasks, Datasets and Baseline System. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)* (pp. 85–92). (Cited on pages xvii, 9, 22, 23, 26, and 41.)

Mesaros, A., Heittola, T., & Ellis, D. (2018b). *Datasets and Evaluation*, (pp. 147–179). Springer International Publishing: Cham. `https://doi.org/10.1007/978-3-319-63450-0_6`. (Cited on page 27.)

Mesaros, A., Heittola, T., & Virtanen, T. (2016a). Metrics for polyphonic sound event detection. *Applied Sciences*, 6(162). `https://doi.org/10.3390/app6060162`. (Cited on pages 9, 28, 40, and 47.)

Mesaros, A., Heittola, T., & Virtanen, T. (2016b). TUT database for acoustic scene classification and sound event detection. In *24rd European Signal Processing Conference 2016* Budapest, Hungary: IEEE. `https://doi.org/10.1109/EUSIPCO.2016.7760424`. (Cited on pages 2, 18, 26, 28, and 72.)

Mesaros, A., Heittola, T., & Virtanen, T. (2018c). A multi-device dataset for urban acoustic scene classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)* (pp. 9–13). Surrey, UK. (Cited on pages 9, 27, and 41.)

Messner, E., Fediuk, M., Swatek, P., Scheidl, S., Smolle-Jüttner, F.-M., Olschewski, H., & Pernkopf, F. (2020). Multi-channel lung sound classification with convolutional recurrent neural networks. *Computers in Biology and Medicine*, 122, 103831. `https://doi.org/10.1016/j.compbiomed.2020.103831`. (Cited on page 2.)

Mishra, S., Benetos, E., Sturm, B. L. T., & Dixon, S. (2020). Reliable Local Explanations for Machine Listening. In *2020 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). Glasgow, UK: IEEE. `https://doi.org/10.1109/IJCNN48605.2020.9207444`. (Cited on page 34.)

Mishra, S., Sturm, B. L., & Dixon, S. (2017). Local Interpretable Model-Agnostic Explanations for Music Content Analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference* (pp. 537–543). Suzhou, China: ISMIR. `https://doi.org/10.5281/zenodo.1417387`. (Cited on page 34.)

Molnar, C. (2019). *Interpretable Machine Learning.* `https://christophm.github.io/interpretable-ml-book/`. (Cited on pages 1, 2, and 5.)

Monarch, R. M. (2021). *Human-in-the-Loop Machine Learning.* Manning. (Cited on page 102.)

Morrison, M. & Pardo, B. (2019). OtoMechanic: Auditory Automobile Diagnostics via Query-by-Example. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)* (pp. 169–173). New York University, NY, USA. (Cited on page 2.)

Muckenhirn, H., Abrol, V., Magimai-Doss, M., & Marcel, S. (2019). Understanding and Visualizing Raw Waveform-Based CNNs. In *Proceedings of INTERSPEECH 2019* (pp. 2345–2349). Graz, Austria: ISCA. http://doi.org/10.21437/Interspeech.2019-2341. (Cited on page 34.)

Mulimani, M. & Koolagudi, S. G. (2016). *Acoustic Scene Classification Using MFCC and MP Features.* Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 20.)

Mydlarz, C., Nacach, S., Roginska, A., Park, T. H., Rosenthal, E., & Temple, M. (2014). The Implementation of MEMS Microphones for Urban Sound Sensing. In *Audio Engineering Society Convention 137*. http://www.aes.org/e-lib/browse.cfm?elib=17466. (Cited on page 2.)

Mydlarz, C., Shamoon, C., Baglione, M., & Pimpinella, M. (2015). The design and calibration of low cost urban acoustic sensing devices. In *EuroNoise 2015* (pp. 2345–2350).: European Acoustics Association. (Cited on page 2.)

Narayanaswamy, S., et al. (2017). Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017,* (pp. 5925–5935). Long Beach, CA, USA. https://proceedings.neurips.cc/paper/2017/hash/9cb9ed4f35cf7c2f295cc2bc6f732a84-Abstract.html. (Cited on page 6.)

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). Synthesizing the Preferred Inputs for Neurons in Neural Networks via Deep Generator Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16 (pp. 3395–3403). Red Hook, NY, USA: Curran Associates Inc. https://proceedings.neurips.cc/paper/2016/hash/5d79099fcdf499f12b79770834c0164a-Abstract.html. (Cited on page 31.)

Nickerson, R. S. (1998). Confirmation Bias: A Ubiquitous Phenomenon in Many Guises. *Review of General Psychology*, 2(2), 175–220. https://doi.org/10.1037/1089-2680.2.2.175. (Cited on page 99.)

Nolasco, I., et al. (2022). Few-shot bioacoustic event detection at the DCASE 2022 challenge. *CoRR*, abs/2207.07911. `https://doi.org/10.48550/arXiv.2207.07911`. (Cited on page 36.)

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., & Mordvintsev, A. (2018). The Building Blocks of Interpretability. *Distill*. `https://doi.org/10.23915/distill.00010`. (Cited on page 103.)

Ooi, K., et al. (2021). A Strongly-Labelled Polyphonic Dataset of Urban Sounds with Spatiotemporal Context. In *Proceedings of the 13th Asia Pacific Signal and Information Processing Association Annual Summit and Conference* (pp. 982–988). Tokyo, Japan: IEEE. `https://ieeexplore.ieee.org/document/9689561`. (Cited on page 26.)

Pajusco, N., Huang, R., & Farrugia, N. (2020). Lightweight Convolutional Neural Networks on Binaural Waveforms for Low Complexity Acoustic Scene Classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)* (pp. 135–139). Tokyo, Japan. (Cited on page 24.)

Pan, Y., et al. (2020). Acoustic Feature Extraction with Interpretable Deep Neural Network for Neurodegenerative Related Disorder Classification. In *Proceedings of INTERSPEECH 2020* (pp. 4806–4810). Shanghai, China: ISCA. `http://doi.org/10.21437/Interspeech.2020-2684`. (Cited on pages 8, 36, and 53.)

Parekh, J., Parekh, S., Mozharovskyi, P., d'Alché-Buc, F., & Richard, G. (2022). Listen to Interpret: Post-hoc Interpretability for Audio Networks with NMF. *CoRR*, abs/2202.11479. `https://arxiv.org/abs/2202.11479`. (Cited on page 35.)

Pelillo, M. & Scantamburlo, T. (2013). How Mature Is the Field of Machine Learning? In M. Baldoni, C. Baroglio, G. Boella, & R. Micalizio (Eds.), *AI*IA 2013: Advances in Artificial Intelligence - XIIIth International Conference of the Italian Association for Artificial Intelligence* (pp. 121–132). Turin, Italy: Springer International Publishing. `https://doi.org/10.1007/978-3-319-03524-6_11`. (Cited on page 3.)

Peng, J., Qu, X., Wang, J., Gu, R., Xiao, J., Burget, L., & Černocký, J. (2021). ICSpk: Interpretable Complex Speaker Embedding Extractor from Raw Waveform. In *Proceedings of Interspeech 2021* (pp. 511–515). `https://doi.org/10.21437/Interspeech.2021-2016`. (Cited on pages 36 and 53.)

Perez-Castanos, S., Naranjo-Alcazar, J., Zuccarello, P., & Cobos, M. (2020). Anomalous Sound Detection using Unsupervised and Semi-Supervised Autoencoders and Gammatone Audio Representation. In *Proceedings of the*

Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020) (pp. 145–149). Tokyo, Japan. (Cited on page 24.)

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 2227–2237). New Orleans, LA, USA: Association for Computational Linguistics. `https://doi.org/10.18653/v1/N18-1202`. (Cited on page 1.)

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd ACM international conference on Multimedia* (pp. 1015–1018). Brisbane, Australia. `https://doi.org/10.1145/2733373.2806390`. (Cited on pages 9 and 41.)

Politis, A., Mesaros, A., Adavanne, S., Heittola, T., & Virtanen, T. (2020). Overview and Evaluation of Sound Event Localization and Detection in DCASE 2019. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 684–698. `https://doi.org/10.1109/TASLP.2020.3047233`. (Cited on page 101.)

Pons, J., Lidy, T., & Serra, X. (2016). Experimenting with musically motivated convolutional neural networks. In *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)* (pp. 1–6). Bucharest, Romania: IEEE. `https://doi.org/10.1109/CBMI.2016.7500246`. (Cited on page 70.)

Pons, J., Nieto, O., Prockup, M., Schmidt, E., Ehmann, A., & Serra, X. (2018). End-to-end Learning for Music Audio Tagging at Scale. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*. `https://doi.org/10.5281/zenodo.1492497`. (Cited on page 53.)

Pons, J., Serrà, J., & Serra, X. (2019). Training Neural Audio Classifiers with Few Data. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019* (pp. 16–20). Brighton, United Kingdom: IEEE. `https://doi.org/10.1109/ICASSP.2019.8682591`. (Cited on page 36.)

Pons, J. & Serra, X. (2019). Randomly Weighted CNNs for (Music) Audio Classification. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 336–340). Brighton, United Kingdom: IEEE. `https://doi.org/10.1109/ICASSP.2019.8682912`. (Cited on page 102.)

Purohit, T. & Agarwal, A. (2018). *Acoustic Scene Classification Using Deep CNN on Raw-Waveform*. Technical report, Detection and Classification of Acoustic Scenes and Events 2018 Challenge. (Cited on page 24.)

Qian, K., Ren, Z., Pandit, V., Yang, Z., Zhang, Z., & Schuller, B. (2017). Wavelets Revisited for the Classification of Acoustic Scenes. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)* (pp. 108–112). (Cited on page 20.)

Qin, Y., Carlini, N., Cottrell, G., Goodfellow, I., & Raffel, C. (2019). Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research* (pp. 5231–5240). Long Beach, CA, USA: PMLR. `http://proceedings.mlr.press/v97/qin19a.html`. (Cited on page 1.)

Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., & Ellis, D. P. W. (2014). mir_eval: A Transparent Implementation of Common MIR Metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference* (pp. 367–372). Taipei, Taiwan. `https://doi.org/10.5281/zenodo.1416528`. (Cited on page 8.)

Ramaswamy, V. V., Kim, S. S. Y., Meister, N., Fong, R., & Russakovsky, O. (2022). ELUDE: Generating interpretable explanations via a decomposition into labelled and unlabelled features. *CoRR*, abs/2206.07690. `https://doi.org/10.48550/arXiv.2206.07690`. (Cited on pages 32 and 102.)

Ramires, A., Chandna, P., Favory, X., Gómez, E., & Serra, X. (2020). Neural Percussive Synthesis Parameterised by High-Level Timbral Features. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 786–790). Barcelona, Spain: IEEE. `https://doi.org/10.1109/ICASSP40776.2020.9053128`. (Cited on page 101.)

Ras, G., Xie, N., van Gerven, M., & Doran, D. (2022). Explainable Deep Learning: A Field Guide for the Uninitiated. *Journal of Artificial Intelligence Research (JAIR)*, 73, 329–396. `https://doi.org/10.1613/jair.1.13200`. (Cited on pages 1, 5, and 6.)

Ravanelli, M. & Bengio, Y. (2018). Speaker Recognition from Raw Waveform with SincNet. In *2018 IEEE Spoken Language Technology Workshop (SLT)* (pp. 1021–1028). `https://doi.org/10.1109/SLT.2018.8639585`. (Cited on pages 36 and 53.)

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16 (pp. 1135–1144). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778. (Cited on pages 6 and 29.)

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, (pp. 65–386). (Cited on page 21.)

Ross, A. S., Hughes, M. C., & Doshi-Velez, F. (2017). Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (pp. 2662–2670). https://doi.org/10.24963/ijcai.2017/371. (Cited on page 32.)

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1, 206–215. https://doi.org/10.1038/s42256-019-0048-x. (Cited on pages 1, 5, 6, and 95.)

Sacha, D., et al. (2017). What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, 268, 164–175. Advances in artificial neural networks, machine learning and computational intelligence. https://doi.org/10.1016/j.neucom.2017.01.105. (Cited on page 103.)

Sainath, T. N., et al. (2017). *Raw Multichannel Processing Using Deep Neural Networks*, (pp. 105–133). Springer International Publishing. https://doi.org/10.1007/978-3-319-64680-0_5. (Cited on page 53.)

Sakashita, Y. & Aono, M. (2018). *Acoustic Scene Classification by Ensemble of Spectrograms Based on Adaptive Temporal Divisions*. Technical report, Detection and Classification of Acoustic Scenes and Events 2018 Challenge. (Cited on page 24.)

Salamon, J. & Bello, J. P. (2015a). Feature learning with deep scattering for urban sound analysis. In *2015 23rd European Signal Processing Conference (EUSIPCO)* (pp. 724–728). https://doi.org/10.1109/EUSIPCO.2015.7362478. (Cited on page 18.)

Salamon, J. & Bello, J. P. (2015b). Unsupervised feature learning for urban sound classification. In *2015 International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 171–175).: IEEE. https://doi.org/10.1109/ICASSP.2015.7177954. (Cited on pages 3 and 18.)

Salamon, J. & Bello, J. P. (2017). Deep Convolutional Neural Networks and Data Augmentation For Environmental Sound Classification. *IEEE Signal Processing Letters*, 24, 279–283. https://doi.org/10.1109/LSP.2017.2657381. (Cited on pages xvii, 9, 23, 24, 25, 46, 47, and 74.)

Salamon, J., Jacoby, C., & Bello, J. P. (2014). A Dataset and Taxonomy for Urban Sound Research. In K. A. Hua, Y. Rui, R. Steinmetz, A. Hanjalic, A. Natsev, & W. Zhu (Eds.), *Proceedings of the ACM International Conference on Multimedia* (pp. 1041–1044). Orlando, FL, USA: ACM. https://doi.org/10.1145/2647868.2655045. (Cited on pages 3, 9, 20, 26, 27, 41, and 73.)

Salamon, J., MacConnell, D., Cartwright, M., Li, P., & Bello., J. P. (2017). Scaper: A Library for Soundscape Synthesis and Augmentation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* New York, USA. https://doi.org/10.1109/WASPAA.2017.8170052. (Cited on pages 9, 17, 18, 26, 27, 41, 46, 47, 54, 56, 58, 59, and 91.)

Sato, M. & Tsukimoto, H. (2001). Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, volume 3 (pp. 1870–1875 vol.3). https://doi.org/10.1109/IJCNN.2001.938448. (Cited on page 30.)

Schedl, M. (2019). Deep Learning in Music Recommendation Systems. *Frontiers in Applied Mathematics and Statistics*, 5, 44. https://doi.org/10.3389/fams.2019.00044. (Cited on page 1.)

Schiller, D., Huber, T., Lingenfelser, F., Dietz, M., Seiderer, A., & André, E. (2019). Relevance-Based Feature Masking: Improving Neural Network Based Whale Classification Through Explainable Artificial Intelligence. In *Proceedings of INTERSPEECH 2019* (pp. 2423–2427). Graz, Austria: ISCA. http://doi.org/10.21437/Interspeech.2019-2707. (Cited on page 34.)

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 618–626). Venice, Italy: IEEE Computer Society. https://doi.org/10.1109/ICCV.2017.74. (Cited on page 29.)

Sena Mafra, G., Duong, Q.-K.-N., Ozerov, A., & Perez, P. (2016). *Acoustic Scene Classification: an Evaluation of an Extremely Compact Feature Representation.* Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 20.)

Serizel, R., Bisot, V., Essid, S., & Richard, G. (2018). *Acoustic Features for Environmental Sound Analysis*, (pp. 71–101). Springer International Publishing: Cham. https://doi.org/10.1007/978-3-319-63450-0_4. (Cited on page 20.)

Serizel, R., Turpault, N., Shah, A., & Salamon, J. (2020). Sound Event Detection in Synthetic Domestic Environments. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 86–90). Barcelona, Spain: IEEE. https://doi.org/10.1109/ICASSP40776.2020.9054478. (Cited on page 27.)

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning Important Features Through Propagating Activation Differences. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research* (pp. 3145–3153).: PMLR. https://proceedings.mlr.press/v70/shrikumar17a.html. (Cited on page 29.)

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *International Conference on Learning Representations (ICLR)* Banff, Canada: ICLR. http://arxiv.org/abs/1312.6034. (Cited on pages 6, 28, and 30.)

Simonyan, K. & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1409.1556. (Cited on page 53.)

Six, J., Bressan, F., & Leman, M. (2018). A case for reproduciblity in MIR: replication of 'a highly robust audio fingerprinting system'. *Transactions of the International Society for Music Information Retrieval*, 1(1), 56–67. http://dx.doi.org/10.5334/tismir.4. (Cited on pages 9 and 39.)

Slade, S. (1991). Case-Based Reasoning: A Research Paradigm. *AI Magazine*, 12(1), 42. https://doi.org/10.1609/aimag.v12i1.883. (Cited on page 32.)

Snell, J., Swersky, K., & Zemel, R. S. (2017). Prototypical Networks for Few-shot Learning. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017* (pp. 4077–4087). Long Beach, CA, USA. https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html. (Cited on page 36.)

Springenberg, J., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for Simplicity: The All Convolutional Net. In *ICLR (workshop track)*. http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a. (Cited on page 29.)

Stowell, D. (2018). *Computational Bioacoustic Scene Analysis*, (pp. 303–333). Springer International Publishing: Cham. https://doi.org/10.1007/978-3-319-63450-0_11. (Cited on page 2.)

Sun, J., Lapuschkin, S., Samek, W., Zhao, Y., Cheung, N., & Binder, A. (2020). Explanation-Guided Training for Cross-Domain Few-Shot Classification. In *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021* (pp. 7609–7616).: IEEE, doi:10.1109/ICPR48806.2021.9412941. https://doi.org/10.1109/ICPR48806.2021.9412941. (Cited on page 103.)

Sun, S., Sun, Q., Zhou, K., & Lv, T. (2019). Hierarchical Attention Prototypical Networks for Few-Shot Text Classification. In *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 476–485). Hong Kong, China. https://doi.org/10.18653/v1/D19-1045. (Cited on page 33.)

Szegedy, C., et al. (2015). Going Deeper with Convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Boston, MA, USA. https://doi.org/10.1109/CVPR.2015.7298594. (Cited on page 53.)

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations* Banff, Canada: ICLR. http://arxiv.org/abs/1312.6199. (Cited on page 1.)

Sánchez-Hevia, H. A., Ayllón, D., Gil-Pita, R., & Rosa-Zurera, M. (2017). Maximum Likelihood Decision Fusion for Weapon Classification in Wireless Acoustic Sensor Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6), 1172–1182. https://doi.org/10.1109/TASLP.2017.2690579. (Cited on page 2.)

Tanabe, R., Purohit, H., Dohi, K., Endo, T., Nikaido, Y., Nakamura, T., & Kawaguchi, Y. (2021). MIMII Due: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection with Domain Shifts Due to Changes in Operational and Environmental Conditions. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2021* (pp. 21–25).: IEEE. https://doi.org/10.1109/WASPAA52581.2021.9632802. (Cited on page 27.)

Tax, T. M. S., Antich, J. L. D., Purwins, H., & Maaløe, L. (2017). Utilizing Domain Knowledge in End-to-End Audio Processing. In *31st Conference on Neural Information Processing Systems (NIPS)* Long Beach, CA, USA. http://arxiv.org/abs/1712.00254. (Cited on pages 8, 36, 46, 53, 54, and 56.)

Thickstun, J., Harchaoui, Z., & Kakade, S. M. (2017). Learning Features of Music from Scratch. In *International Conference on Learning Representations* Toulon, France: ICLR. https://openreview.net/forum?id=rkFBJv9gg. (Cited on pages 8, 36, and 53.)

Tokozume, Y. & Harada, T. (2017). Learning environmental sounds with end-to-end convolutional neural network. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2721–2725). New Orleans, LA, USA. https://doi.org/10.1109/ICASSP.2017.7952651. (Cited on page 36.)

Tolan, S., Miron, M., Gómez, E., & Castillo, C. (2019). Why Machine Learning May Lead to Unfairness: Evidence from Risk Assessment for Juvenile Justice in Catalonia. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*, ICAIL '19 (pp. 83–92). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3322640.3326705. (Cited on page 4.)

Turian, J., et al. (2022). HEAR 2021: Holistic Evaluation of Audio Representations. *CoRR*, abs/2203.03022. https://doi.org/10.48550/arXiv.2203.03022. (Cited on page 102.)

Tüske, Z., Golik, P., Schlüter, R., & Ney, H. (2014). Acoustic modeling with deep neural networks using raw time signal for LVCSR. In *Proceedings of INTERSPEECH 2014* (pp. 890–894). Singapore: ISCA. http://www.isca-speech.org/archive/interspeech_2014/i14_0890.html. (Cited on page 53.)

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2020). Deep Image Prior. *International Journal of Computer Vision*, 128, 1867–1888. https://doi.org/10.1007/s11263-020-01303-4. (Cited on page 31.)

Valenti, G., Daniel, A., & Evans, N. (2018). End-to-end automatic speaker verification with evolving recurrent neural networks. In *Proceedings of Odyssey 2018 The Speaker and Language Recognition Workshop* (pp. 335–341). https://doi.org/10.21437/Odyssey.2018-47. (Cited on page 53.)

Valero, X. & Alias, F. (2012). Gammatone Cepstral Coefficients: Biologically Inspired Features for Non-Speech Audio Classification. *IEEE Transactions on Multimedia*, 14(6), 1684–1689. https://doi.org/10.1109/TMM.2012.2199972. (Cited on page 18.)

Vandewalle, P., Kovacevic, J., & Vetterli, M. (2009). Reproducible research in signal processing. *IEEE Signal Processing Magazine*, 26(3), 37–47. https://doi.org/10.1109/MSP.2009.932122. (Cited on page 8.)

Vaswani, A., et al. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 30: Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. (Cited on page 32.)

Vikaskumar, G., Waldekar, S., Paul, D., & Saha, G. (2016). *Acoustic Scene Classification Using Block Based MFCC Features.* Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 20.)

Virtanen, T., Plumbley, M. D., & Ellis, D. (2018). *Introduction to Sound Scene and Event Analysis*, (pp. 3–12). Springer International Publishing: Cham. https://doi.org/10.1007/978-3-319-63450-0_1. (Cited on page 3.)

Vu, T. H. & Wang, J.-C. (2016). *Acoustic Scene and Eevent Recognition using Recurrent Neural Networkds.* Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 17.)

Waldekar, S. & Saha, G. (2018). *Wavelet-Based Audio Features for Acoustic Scene Classification.* Technical report, Detection and Classification of Acoustic Scenes and Events 2018 Challenge. (Cited on page 20.)

Wang, C.-H., You, J.-K., & Liu, Y.-W. (2017a). *Sound Event Detection From Real-Life Audio by Training a Long Short-Term Memory Network with Mono and Stereo Features.* Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 18.)

Wang, Y., Getreuer, P., Hughes, T., Lyon, R. F., & Saurous, R. A. (2017b). Trainable frontend for robust and far-field keyword spotting. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5670–5674). New Orleans, LA, USA: IEEE. https://doi.org/10.1109/ICASSP.2017.7953242. (Cited on pages 18, 19, and 57.)

Wang, Z., Wang, D., Zhang, Y., & Xia, G. (2020). Learning Interpretable Representation for Controllable Polyphonic Music Generation. In *Proceedings of the 21st International Society for Music Information Retrieval Conference* (pp. 662–669). Montréal, Canada: ISMIR, doi:https://doi.org/10.5281/zenodo.4245518. (Cited on page 35.)

Warden, P. (2018). Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *CoRR*, abs/1804.03209. http://arxiv.org/abs/1804.03209. (Cited on page 74.)

Wei, D., Li, J., Pham, P., Das, S., Qu, S., & Metze, F. (2016). *Sound Event Detection for Real Life Audio DCASE Challenge.* Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 18.)

Wiegreffe, S. & Pinter, Y. (2019). Attention is not not Explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 11–20). Hong Kong, China: Association for Computational Linguistics. `https://doi.org/10.18653/v1/D19-1002`. (Cited on pages 6 and 32.)

Witten, I. H. & Frank, E. (2005). Credibility: Evaluating What's Been Learned. In *Data Mining. Practical Machine Learning Tools and Techniques* chapter 5, (pp. 143–185). Elsevier, 2 edition. (Cited on page 43.)

Won, M., Chun, S., Nieto, O., & Serrc, X. (2020). Data-Driven Harmonic Filters for Audio Representation Learning. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 536–540). Barcelona, Spain: IEEE. `https://doi.org/10.1109/ICASSP40776.2020.9053669`. (Cited on pages 8, 37, and 53.)

Won, M., Chun, S., & Serra, X. (2019). Toward Interpretable Music Tagging with Self-Attention. *CoRR*, abs/1906.04972. `http://arxiv.org/abs/1906.04972`. (Cited on pages 32 and 36.)

Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional Block Attention Module. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (pp. 3–19). Cham: Springer International Publishing. `https://doi.org/10.1007/978-3-030-01234-2_1`. (Cited on page 32.)

Wu, H.-H., Seetharaman, P., Kumar, K., & Bello, J. P. (2022). Wav2CLIP: Learning Robust Audio Representations from Clip. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4563–4567). Singapore: IEEE. `https://doi.org/10.1109/ICASSP43922.2022.9747669`. (Cited on page 102.)

Xu, T., White, J. A., Kalkan, S., & Gunes, H. (2020). Investigating Bias and Fairness in Facial Expression Recognition. In *16th European Conference on Computer Vision (ECCV 2020)*, volume 12540 (pp. 506–523). `https://doi.org/10.17863/CAM.56933`. (Cited on page 4.)

Yang, S., Liu, A. T., & Lee, H. (2020). Understanding Self-Attention of Self-Supervised Audio Transformers. In *Proceedings of INTERSPEECH 2020* (pp. 3785–3789). Shanghai, China: ISCA. `https://doi.org/10.21437/Interspeech.2020-2231`. (Cited on page 36.)

Yeom, S.-K., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K.-R., & Samek, W. (2021). Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115, 107899. `https://doi.org/10.1016/j.patcog.2021.107899`. (Cited on page 103.)

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*, volume 27 (pp. 3320–3328).: Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2014/file/375c71349b295fbe2dcdca9206f20a06-Paper.pdf`. (Cited on pages 6 and 31.)

Zarlenga, M. E., Shams, Z., & Jamnik, M. (2021). Efficient Decompositional Rule Extraction for Deep Neural Networks. *CoRR*, abs/2111.12628. `https://arxiv.org/abs/2111.12628`. (Cited on page 30.)

Zeghidour, N., Usunier, N., Synnaeve, G., Collobert, R., & Dupoux, E. (2018). End-to-End Speech Recognition from the Raw Waveform. In *Proceedings of INTERSPEECH 2018* (pp. 781–785). Hyderabad, India: ISCA. `https://doi.org/10.21437/Interspeech.2018-2414`. (Cited on page 53.)

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701. `http://arxiv.org/abs/1212.5701`. (Cited on page 22.)

Zeiler, M. D. & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 818–833). Cham: Springer International Publishing. `https://doi.org/10.1007/978-3-319-10590-1_53`. (Cited on pages 29 and 30.)

Zhang, Q., Wang, W., & Zhu, S.-C. (2018). Examining CNN Representations With Respect to Dataset Bias. In *AAAI Conference on Artificial Intelligence* (pp. 4464–4473). New Orleans, LA, USA: AAAI Press. `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17429`. (Cited on page 1.)

Zhang, Q., Yang, Y., Ma, H., & Wu, Y. (2019). Interpreting CNNs via Decision Trees. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6254–6263). Los Alamitos, CA, USA: IEEE Computer Society. `https://doi.org/10.1109/CVPR.2019.00642`. (Cited on page 30.)

Zhang, Q.-s. & Zhu, S.-c. (2018). Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1), 27–39. `https://doi.org/10.1631/FITEE.1700808`. (Cited on pages 2 and 6.)

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning Deep Features for Discriminative Localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2921–2929). Los Alamitos, CA, USA: IEEE Computer Society. `https://doi.org/10.1109/CVPR.2016.319`. (Cited on page 29.)

Zhou, B., Sun, Y., Bau, D., & Torralba, A. (2018). Interpretable Basis Decomposition for Visual Explanation. In *European Conference on Computer Vision*, volume 11212 (pp. 122–138). `https://doi.org/10.1007/978-3-030-01237-3_8`. (Cited on page 31.)

Zhou, J. (2017). *Sound Event Detection in Multichannel Audio LSTM Network*. Technical report, Detection and Classification of Acoustic Scenes and Events 2017 Challenge. (Cited on page 17.)

Zhu, X. & Bain, M. (2017). B-CNN: Branch Convolutional Neural Network for Hierarchical Classification. *CoRR*, abs/1709.09890. `http://arxiv.org/abs/1709.09890`. (Cited on page 59.)

Zilke, J. R., Loza Mencía, E., & Janssen, F. (2016). DeepRED – Rule Extraction from Deep Neural Networks. In T. Calders, M. Ceci, & D. Malerba (Eds.), *Discovery Science* (pp. 457–473). Cham: Springer International Publishing. `https://doi.org/10.1007/978-3-319-46307-0_29`. (Cited on page 30.)

Zinemanas, P., Cancela, P., & Rocamora, M. (2019a). End-to-end Convolutional Neural Networks for Sound Event Detection in Urban Environments. In *24th Conference of Open Innovations Association* (pp. 533–539). Moscow, Russia: FRUCT IEEE. `https://doi.org/10.23919/FRUCT.2019.8711906`. (Cited on pages 46, 47, and 98.)

Zinemanas, P., Cancela, P., & Rocamora, M. (2019b). MAVD: a dataset for sound event detection in urban environments. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)* New York, USA. (Cited on pages 3, 9, 10, 26, 41, 47, and 99.)

Zinemanas, P., Hounie, I., Cancela, P., Font, F., Rocamora, M., & Serra, X. (2020). DCASE-models: a Python library for computational environmental sound analysis using deep-learning models. In *Proceedings of the Fifth Workshop on Detection and Classification of Acoustic Scenes and Events* (pp. 240–244). Tokyo, Japan: DCASE. (Cited on pages 3, 76, 91, and 98.)

Zinemanas, P., Rocamora, M., Fonseca, E., Font, F., & Serra, X. (2021a). Toward Interpretable Polyphonic Sound Event Detection with Attention Maps Based on Local Prototypes. In *Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)* (pp.

50–54). Barcelona, Spain. https://doi.org/10.5281/zenodo.5770113. (Cited on page 99.)

Zinemanas, P., Rocamora, M., Miron, M., Font, F., & Serra, X. (2021b). An Interpretable Deep Learning Model for Automatic Sound Classification. *Electronics*, 10(7). https://doi.org/10.3390/electronics10070850. (Cited on pages 3 and 99.)

Zöhrer, M. & Pernkopf, F. (2016). *Gated Recurrent Networks Applied to Acoustic Scene Classification and Acoustic Event Detection.* Technical report, Detection and Classification of Acoustic Scenes and Events 2016 Challenge. (Cited on page 17.)

Appendix B: publications by the author

In press

Fuentes, M., Salamon, J., Zinemanas, P., Rocamora, M., Paja, G., Román, I.R., Bittner, R.M., Miron, M., Serra, X., and Bello, J.P. Soundata: A Python library for reproducible use of audio datasets. *ArXiv, abs/2109.12690*, September 2021.

Journal papers

Zinemanas, P., Rocamora, M., Miron, M., Font, F., Serra, X. An Interpretable Deep Learning Model for Automatic Sound Classification. *Electronics 10*(7:850) 2021, https://doi.org/10.3390/electronics10070850

Conference papers

Singh, R., Zinemanas, P., Serra, X., Bello, J.P., Fuentes, M. Flowgrad: Using Motion for Visual Sound Source Localization. *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023), June 2023*

Fuentes, M., Steers, B., Zinemanas, P., Rocamora, M., Bondi, L., Wilkins, J., Shi, Q., Hou, Y., Das, S., Serra, X., Bello, J.P. Urban Sound & Sight: Dataset And Benchmark For Audio-Visual Urban Scene Understanding. *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2022), May 2022*

Zinemanas, P., Rocamora, M., Fonseca, E., Font, F., Serra X. Toward interpretable polyphonic sound event detection with attention maps based on local prototypes. *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2021)*, November 2021.

Zinemanas, P., Hounie, I., Cancela, P., Font, F., Rocamora, M., Serra X. DCASE-models: A Python library for computational environmental sound analysis using deep-learning models. *5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2020)*, November 2020.

Zinemanas, P., Cancela, P., Rocamora, M. MAVD: a dataset for sound event detection in urban environments. *4th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2019)*, October 2019.

Zinemanas, P., Cancela, P., Rocamora, M. End-to-end convolutional neural networks for sound event detection in urban environments. *24th Conference of Open Innovations Association (FRUCT 2019)*, March 2019.