electronics

MDPI

Article

An Interpretable Deep Learning Model for Automatic Sound Classification

Pablo Zinemanas [1,*], Martín Rocamora [2], Marius Miron [1], Frederic Font [1] and Xavier Serra [1]

[1] Music Technology Group, Universitat Pompeu Fabra, 08018 Barcelona, Spain; marius.miron@upf.edu (M.M.); frederic.font@upf.edu (F.F.); xavier.serra@upf.edu (X.S.)
[2] Facultad de Ingeniería, Universidad de la República, Montevideo 11300, Uruguay; rocamora@fing.edu.uy
* Correspondence: pablo.zinemanas@upf.edu

Abstract: Deep learning models have improved cutting-edge 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. There is still a lack of research in the audio domain, despite the increasing interest in developing deep learning models that provide explanations of their decisions. To reduce this gap, we propose 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 and by considering different time-frequency resolutions in the feature space. The proposed model achieves results that are comparable to that of the state-of-the-art methods in three different sound classification tasks involving speech, music, and environmental audio. In addition, we present two automatic methods to prune the proposed model that exploit its interpretability. Our system is open source and it is accompanied by a web application for the manual editing of the model, which allows for a human-in-the-loop debugging approach.

Keywords: interpretability; explainability; deep learning; sound classification; prototypes

check for updates

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

Academic Editors: Chiman Kwan, Alexander Lerch and Peter Knees

Received: 27 February 2021
Accepted: 31 March 2021
Published: 2 April 2021

1. Introduction

The popularization of deep learning has led to significant advances in a wide range of scientific fields and practical problems [1,2], most notably in computer vision [3] and natural language processing [4], but also in the audio domain, improving the state of the art of several tasks, such as speech recognition [5] and music recommendation [6]. 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 for DNNs to learn useful representations of the data, but, at the same time, 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 [7–9], and makes it difficult to extract the knowledge that is captured by the model about the problem in a way that humans can understand [10]. Particularly, in applications that impact human lives—such as in healthcare—the lack of transparency and accountability can have serious consequences [11]. It can be argued that the difficulty to understand these models is not problematic in many successful applications of DNNs [12,13]. However, prediction with high probability does not guarantee the model to always behave as expected. For instance, it has been shown that DNNs can be fooled with certain ease in classification tasks by adversarial attacks [14]—see [15,16] for examples in speech.

The integration of 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) [17]. 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* [10,18]. Models that can be interpreted can be better debugged and audited, either to devise defense methods, foresee malfunctions, discover edge cases, or detect biases [10,19].

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 [20]. Naturally, this does not mean a low-level mechanistic understanding of the inner workings of the model, but a rationale behind its predictions in a way that the user can interpret [18]. The lack of agreement on the terms is misleading, and it mainly arises from their lack of precise meanings when applied to algorithms [13]. A problematic aspect of interpretability is that it is a domain-specific notion and, therefore, there is no single definition of what attributes render models interpretable [11]. Yet, interpretability is often materialized in practice into a set of application-specific constraints on the model, dictated by domain knowledge, such as causality, monotonicity, or sparsity. In this paper, we propose an intrinsic type of interpretability, based on examples/prototypes, similarly to [21–25]. Our interpretable model should be able to produce explanations in terms of sound prototypes.

Another key issue is that interpretability can be accomplished at different degrees, from a fully transparent to a mildly constrained model [11,18].

In light of the above, it is not surprising that there are a lot of different approaches for explaining DNNs [10,12,18,26]. 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 that are designed to produce explanations or to 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. But 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 [18]. 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 [21,24]. In fact, the ability to interpret the model can reveal the weaknesses of the data or the processing, which can lead to performance improvements if properly addressed.

With respect to 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 proxy linear 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) [27], which learns a linear model as a local approximation of the black-box model in the neighbouring 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 [18]. However, a proxy model is a linear approximation of a non-linear model and it may be an inaccurate representation in some parts of the input space. Thus, the proxy-models are often not reliable [11].

Another type of post-hoc explanation focuses on the representation of the data inside the network. These methods try to explain the role of the layers [28], individual units [19], or other useful representations [29]. Besides, there are visualization methods that highlight the characteristics of the input that strongly influence the output [12]. 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 [30]. They can be useful 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 [31].

For already trained models, post-hoc explanations may be the only option. However, a model that is inherently interpretable provides explanations that are faithful to what the model actually computes [11]. Several different approaches may be taken for designing interpretable or explanation-producing neural networks [18]. For instance, DNNs can be trained to explicitly learn disentangled latent representations [32] 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 as a form of explanation. However, more research is needed to assess whether attention mechanisms can provide reliable explanations [33].

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 [21–25]. Prototype classification is a classical form of case-based reasoning. It stands as an appropriate approach for interpretability that is grounded on how domain experts explain the reasoning processes behind complicated classification tasks in the visual domain. The architecture proposed in [21] appends a special prototype layer to the network. Those prototypes are learned during training and the predictions of the network are based on the similarity of the input to them, being computed as a distance in the latent space. Thus, the explanations that are given by the network are the prototypes and corresponding distances. Subsequently, in [23], this is extended to use hierarchically organized prototypes to classify objects in a predefined taxonomy, and in [24], to learn parts of the training images as prototypes of each class. Finally, the approaches in [22,25] can be regarded as generalizations beyond similarities to prototypes into more general interpretable concepts.

Contributions and Description of This Work

In this work, 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.

Automatic sound classification technologies are becoming widely applied into monitoring systems, smart safety devices, voice assistants, and in different real-world environments, such as industrial, domestic, urban, and natural [34–36]. Consequently, there are various application scenarios in which a human would need to make actionable decisions based on an automatic sound classification system—like surveillance or machine fault monitoring. In such scenarios, humans would benefit from accessing explanations regarding the outputs of the sound classification system. Traditional sound classification systems are based on acoustic features, such as mel-frequency cepstral coefficients (MFCCs), and shallow classifiers, such as Gaussian mixture models (GMMs) [37,38] and decision trees [39]. These are comparatively easier to interpret than classifiers that are based on deep neural networks. However, deep-learning techniques have brought new state-of-the-art results to several sound classification tasks [40–42], thus new research addressing the issue of making these classifiers more interpretable is needed.

In addition, self-explaining networks can be more easily inspected, which has the previously noted advantages.

The proposed network architecture is based on the interpretable model for image classification introduced in [21]. 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 [21]—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 document is organized, as follows. Section 2 reviews previous work in relation to our proposed approach, which is presented in Section 3. Section 4 details the datasets and baseline methods that are used in the experiments reported in Section 5. Section 6 finalizes the paper with the main conclusions and ideas for future work.

2. Relation with Previous Work

In this section, we review 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.

Regarding visualization methods for explainability, saliency maps were applied in [43] 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 [44] for visualizing heat maps in raw waveform convolutional neural networks. It is also possible to propagate the model's prediction backward while using rules about relevance [45] or deep Taylor decomposition [46]. These two methods have been applied to audio classification problems in [47,48], respectively. In contrast to the visualization methods, our explanations come in 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 in [49,50]. It can generate explanations in the form of temporal, frequency, and time-frequency segmentation, and it was applied to singing voice detection. In contrast, our model is designed to be interpretable and we do not generate explanations for black-box models. The time-frequency representations used as input for these models cannot be interpreted as simple images. We argue that example-based explanations are a better fit for these audio representation 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 [51]. The role of each layer in an end-to-end speech recognition systems has been studied in [52]. 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, our network architecture is designed to generate prototypes by reconstructing representations that were learned in a latent space.

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 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 energies in mel-frequency bands [53–55]. Additionally, some works have addressed the design of the first layers of these networks to tailor the feature extraction stage using parametric filters [56–58] or trainable hand-crafted kernels [59,60]. Attention mechanisms have been used to bring interpretability to neural networks in speech and music emotion recognition [61,62] and in music auto-tagging [63]. Recently, a visualization tool was proposed in [64] for understanding the attention mechanisms in self-supervised audio transformers.

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 [65]—in the context of musical instrument recognition—and related to chord and texture [66]—in the context of generative models of polyphonic music. In addition, mid-level features [67] and source separation [68] have been used to improve the model interpretability for the problem of music emotion recognition. In [69], invertible networks [70] have been applied for interpretable automatic polyphonic transcription. Similar to [71], we use domain knowledge to design several parts of the network to learn interpretable representations for speech and audio signals.

Neural network pruning has been shown to reduce the training time and network complexity [72]. To that extent, we refine the proposed model by eliminating redundant prototypes and channels. Thus, we show that our system is able to achieve similar accuracy with a reduced number of parameters.

3. 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.

The diagram shown in Figure 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.

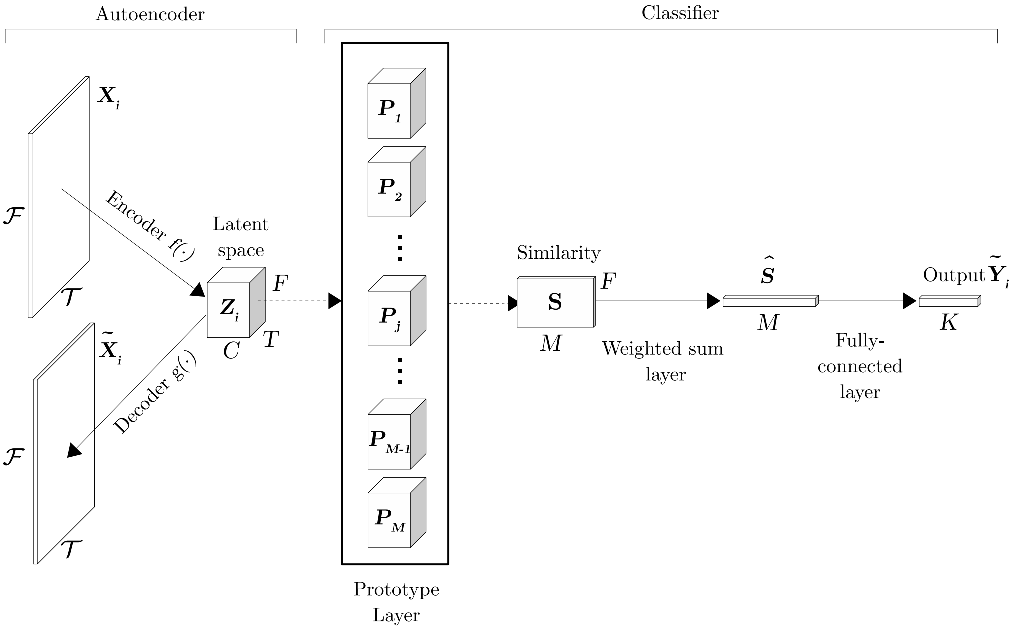

Figure 1. Diagram of the proposed model. Continuous arrows represent layers or sub-networks and they have the name below or to the left of them. Dashed line arrows represent unit connections.

3.1. 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.

3.2. 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,

$$\boldsymbol{Z}_i = f(\boldsymbol{X}_i), \tag{1}$$

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,

$$\tilde{\boldsymbol{X}}_i = g(\boldsymbol{Z}_i) \in \mathbb{R}^{\mathcal{T}\times\mathcal{F}}. \tag{2}$$

The autoencoder that is proposed in [21] 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 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. [73]. 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.

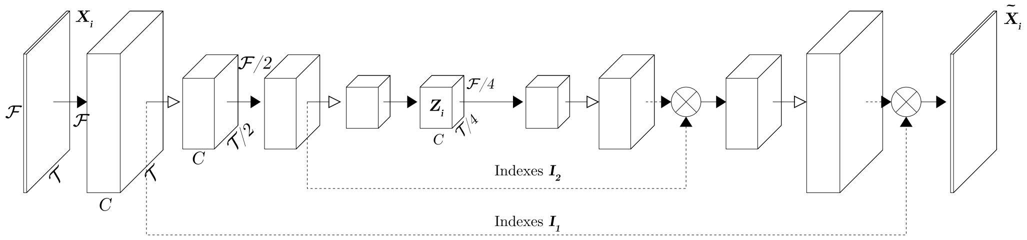

Figure 2. Diagram of the proposed autoencoder. Filled arrows represent convolutional (both normal and transpose) layers. The unfilled arrows point out max-pooling (in the encoder) or upsampling (in the decoder) layers. Dashed line arrows illustrate connections.

A leaky ReLu is the activation after each convolutional layer [74], 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

$$l_r = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{X}_i - \tilde{\boldsymbol{X}}_i\right\|_2^2. \tag{3}$$

3.3. 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 [21]. 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:

$$D_{ij} = \|\boldsymbol{Z}_i - \boldsymbol{P}_j\|_2^2, \tag{4}$$

and then we calculate the following cost function:

$$l_p = \frac{1}{N}\sum_{i=1}^{N}\min_j\{D_{ij}\} + \frac{1}{M}\sum_{j=1}^{M}\min_i\{D_{ij}\}. \tag{5}$$

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 [21]. 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 [21]. 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.

3.4. Similarity Measure and Weighted Sum Layer

Unlike images, the dimensions of the time-frequency representation have different meanings and they should be treated differently [75]. 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:

$$S_{ij}[f] = \exp\left(-\sum_{t=1}^{T}\sum_{c=1}^{C}\left(Z_i[t,f,c] - P_j[t,f,c]\right)^2\right). \tag{6}$$

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:

$$\widehat{S}_{ij} = \sum_{f=1}^{F} H_j[f] S_{ij}[f] \tag{7}$$

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.

3.5. 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:

$$\widetilde{\boldsymbol{Y}} = \text{softmax}(\widehat{\boldsymbol{S}} \cdot \boldsymbol{W}), \tag{8}$$

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, $\boldsymbol{W} = \boldsymbol{I}_K$ [21]. The loss function to train this layer is a categorical cross-entropy,

$$l_c = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} Y_{ik} \log \widetilde{Y}_{ik}. \tag{9}$$

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 3 shows an example of this illustration in the context of a classification task with three classes and using one prototype per class ($M = K = 3$).

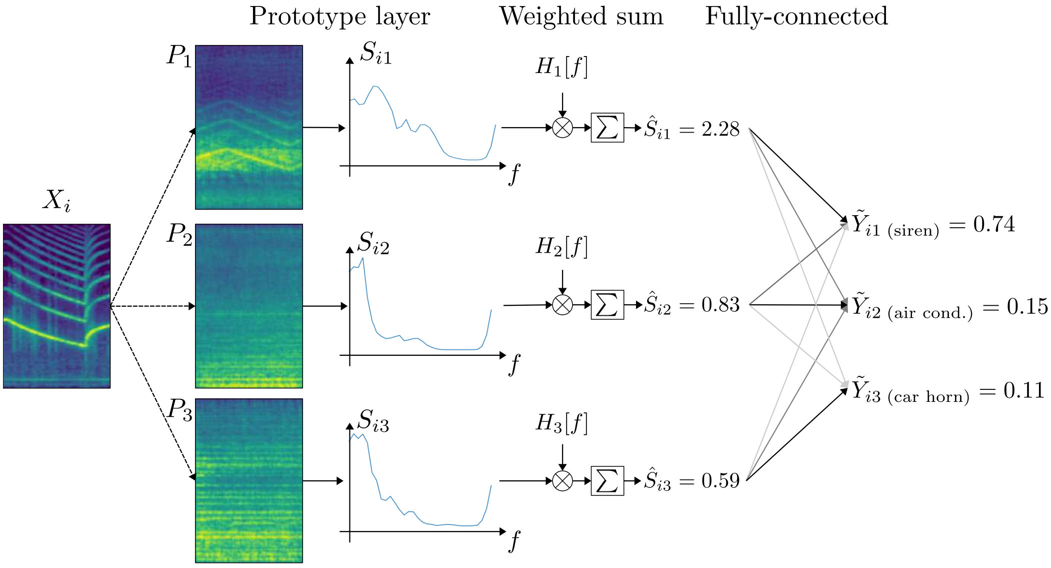

Figure 3. Illustration of how the model makes its predictions. This is an example with three classes: *siren*, *air conditioner*, and *car horn*. The input $\boldsymbol{X_i}$ is compared to three prototypes (one per each class) to get the frequency-dependent similarity $\boldsymbol{S}_{ij}[f]$. This 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.

4. 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.

4.1. 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.

The basic premise of sound classification is that sounds that are produced by the same source of physical process can be grouped into a category. 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 [37].

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.

4.2. Datasets

4.2.1. UrbanSound8k

For the urban sound classification task we use the UrbanSound8K dataset [76]. It has more than 8000 audio slices that are extracted from audio recordings from Freesound [77]. 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.

4.2.2. Medley-Solos-DB

For the music instrument recognition task, we use the Medley-solos-DB dataset [78]. 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 [79] and a test set from the solosDB dataset [80]. The training set also includes a split for model validation.

4.2.3. Google Speech Commands

For the keyword spotting task in speech we use the Google Speech Commands V2 dataset [81]. 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.

4.3. 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) [41]; a convolutional recurrent neural network with an attention mechanism devised for speech classification (Att-CRNN) [42]; and, a pre-trained embedding extractor model based on self-supervised learning of audio-visual data (Openl3) [40].

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-Time 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 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 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**				**Google Speech Commands**			
	fs	m	w, h	l	fs	m	w, h	l	fs	m	w, h	l
APNet	22.05	128	4096, 1024	4.0	44.1	256	4096, 1024	3.0	16.0	80	1024, 256	1.0
SB-CNN	44.1	128	1024, 1024	3.0	44.1	256	1024, 1024	3.0	16.0	80	1024, 256	1.0
Att-CRNN	44.1	128	1024, 1024	3.0	44.1	256	1024, 1024	3.0	16.0	80	1024, 256	1.0
Openl3	48.0	256	512, 242	1.0	48.0	256	512, 242	1.0	48.0	256	512, 242	1.0

4.4. 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 (9). While training APNet, we optimize the weighted sum of all function losses defined previously, Equations (3), (5) and (9):

$$l = \alpha l_c + \beta l_p + \gamma l_r \tag{10}$$

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 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.

Table 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).

	M	C	(α, β, γ)	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

5. 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 models. All of the experiments were conducted using the DCASE-models library [82]. 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 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 3. Mean accuracy (%) and number of parameters (millions) for each model and dataset.

	UrbanSound8K		**Medley-Solos-DB**		**Google Speech Commands**	
	Acc. (%)	**# Params. (M)**	**Acc. (%)**	**# Params. (M)**	**Acc. (%)**	**# Params. (M)**
APNet	76.2	1.2	65.8	4.2	89.0	1.8
SB-CNN	72.2	0.86	64.7	1.8	92.1	0.17
Att-CRNN	61.1	0.23	52.0	0.29	93.2	0.20
Openl3	77.3	9.5	67.3	9.5	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 5.1) and how to refine the network based on the insights provided by the inspection (Section 5.2).

5.1. Network Inspection

In this section, we examine the functioning of the autoencoder (Section 5.1.1) and inspect the learned weights from the prototype (Section 5.1.2), fully-connected (Section 5.1.3), and weighted sum (Section 5.1.4) layers.

5.1.1. 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 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.

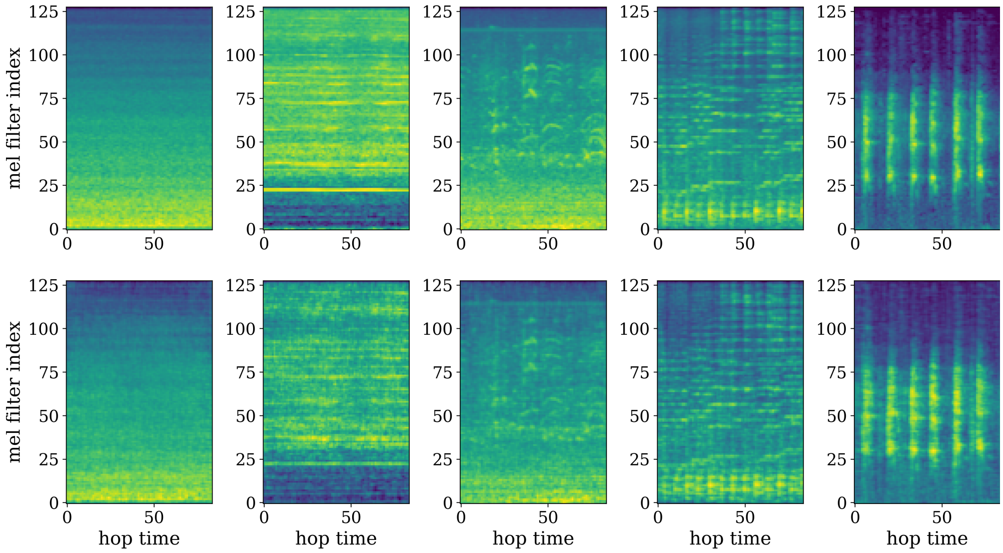

Figure 4. Qualitative assessment of the autoencoder. Random instances from the validation set of the UrbanSound8K dataset in the top row ($\boldsymbol{X}_i$) and their respective reconstructions in the bottom row ($\widetilde{\boldsymbol{X}}_i$).

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 [83] method. This process is implemented while using librosa [84]. The audio signals of the examples of Figure 4 are available in https://pzinemanas.github.io/APNet/.

5.1.2. 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 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/.

5.1.3. 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 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*.

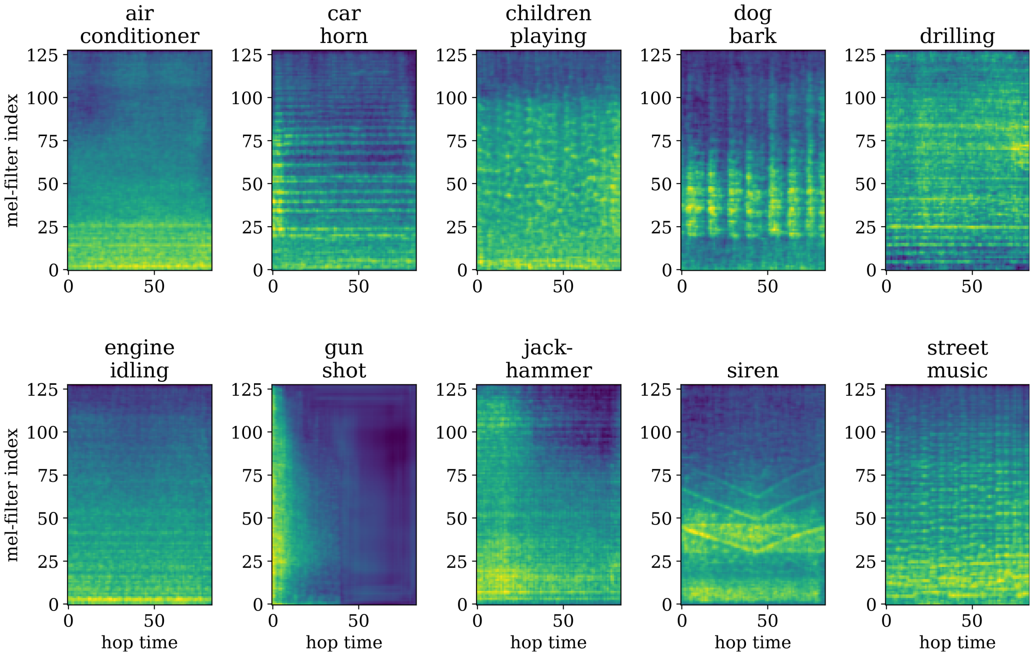

Figure 5. Reconstructed mel-spectrograms of selected prototypes, $g(\boldsymbol{P}_j)$.

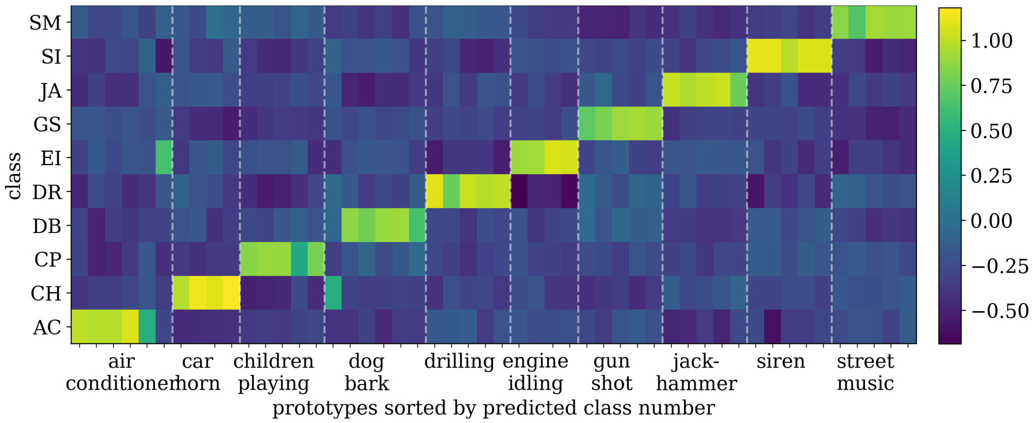

Figure 6. Transposed weight matrix of last fully-connected layer, $\boldsymbol{W}^T$.

5.1.4. 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 7 shows three examples of the learned weights from UrbanSound8K. For instance, in the case of *siren* and *engine idling* prototypes, the weights are 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*.

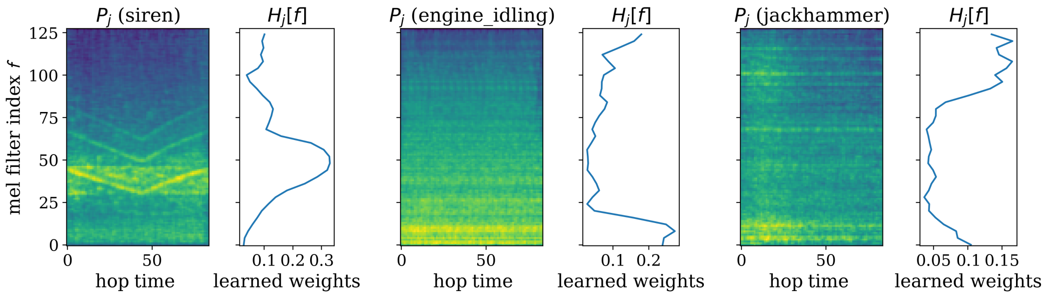

Figure 7. Examples of the learned kernel, $\boldsymbol{H}$, of the weighted sum layer from the similarity function. Note that the weights for each prototype $H_j[f]$ are upsampled to compare them in the input space.

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) replacing the weighted sum layer with a fixed mean operation ($H_j[f] = 1/F \; \forall j, f$). Figure 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.

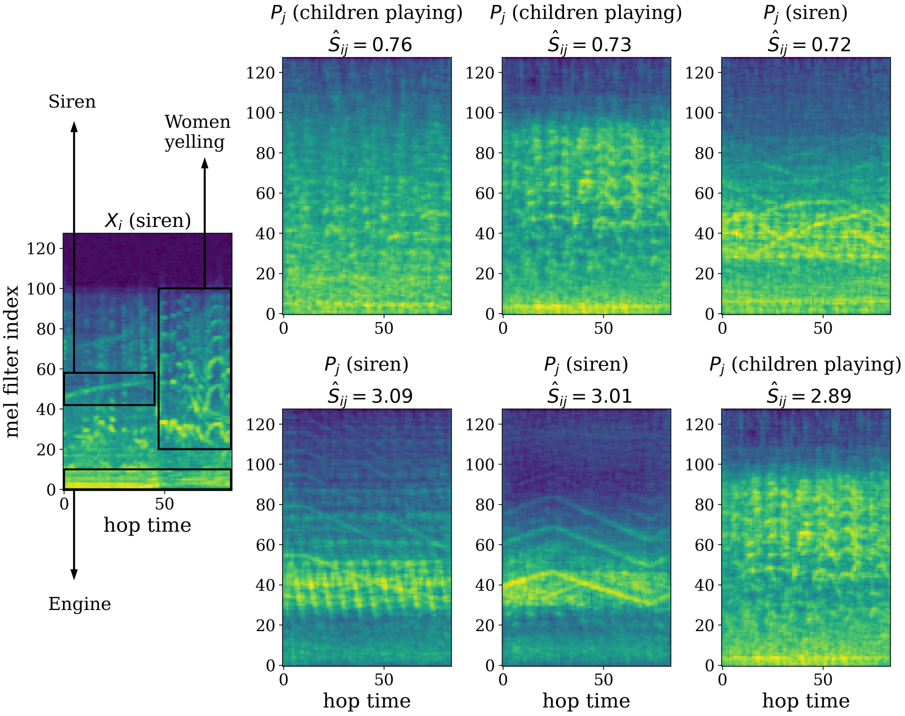

Figure 8. Example on the importance of the weighted sum layer. To the left, the mel-spectrogram, $\boldsymbol{X}_i$, of an audio slice that includes several sources: siren in the background along with a women yelling, and an engine in the foreground. Using a mean operation instead of the weighted sum, 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**).

5.2. 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 5.2.1) and the channels in the encoder's last layer (Section 5.2.2). Besides, we present a web application that is devised for the manual editing of the model (Section 5.2.3).

5.2.1. 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:

$$\mathcal{D}_{jl} = \left\| \boldsymbol{P}_j - \boldsymbol{P}_l \right\|_2^2. \tag{11}$$

Figure 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 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:

$$\mathcal{D}_{jl} < \min\{\mathcal{D}_{jl}\} + \text{mean}\{\mathcal{D}_{jl}\}/2 \quad \forall j,l \in [1,\ldots,M] : j \neq l \tag{12}$$

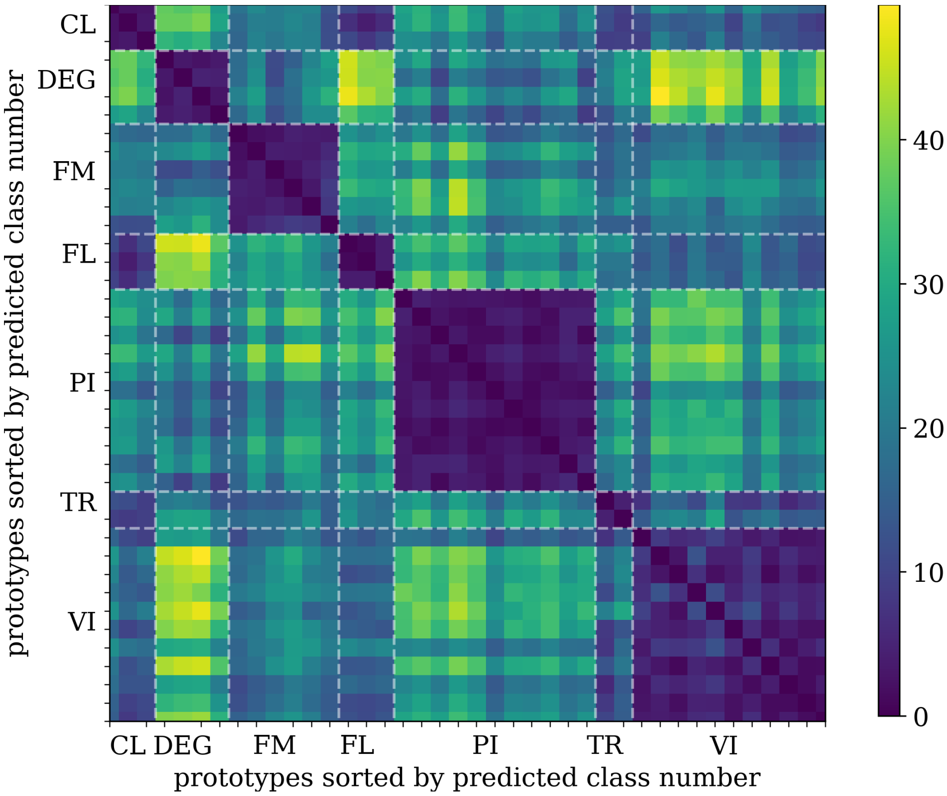

Figure 9. Distance matrix of prototypes of the Medley-solos-DB dataset sorted by predicted classes. The labels are *clarinet* (CL), *distorted electric guitar* (DEG), *female singer* (FM), *flute* (FL), *piano* (PI), *trumpet* (TR), and *violin* (VI).

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. After this network pruning process, we improve the classification results from 65.8% to 68.2%; and, reduce the number of parameters.

5.2.2. 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 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:

$$\mathcal{C}_{kq} = \sum_{i,j,t,f} \left(P_i[t,f,k] - P_j[t,f,q]\right)^2. \tag{13}$$

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 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 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

5.2.3. 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 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.

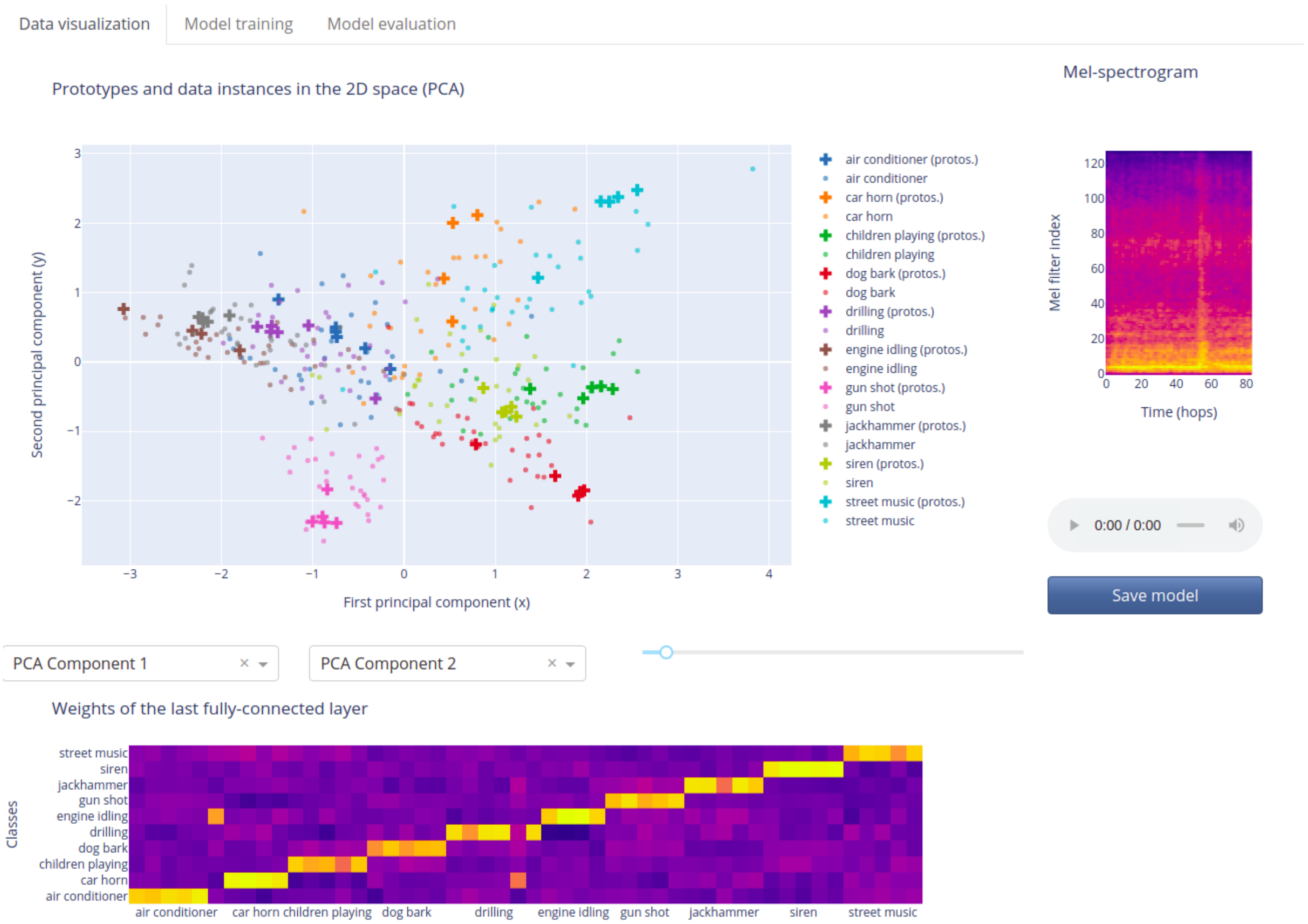

Figure 10. Screenshot of the web application designed for the manual editing of APNet. In this tab the user can navigate through the training set and the prototypes in a two-dimensional (2D) space, listen to the audio files and explore the mel-spectrograms. It is also possible to delete prototypes and convert data instances into prototypes. Other tabs include functions for training and evaluating the edited model.

6. Conclusions

In this work, we present a novel interpretable deep neural network for sound classification—based on an existing model devised for image classification [21]—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.

Future work includes adapting the proposed model to other sound recognition problems, such as sound event detection and audio-tagging. Along with this, we seek to incorporate audio-domain knowledge to the development of other intrinsically interpretable neural network models based on prototypes [23,24] and concepts [22,25]. Besides, we want to use the visualization tool for manual editing to study different ways of training the network with a human-in-the-loop approach; and, to create tools for explaining the inner functionality of the network to end-users.

Author Contributions: Conceptualization, P.Z. and M.R.; methodology, P.Z.; software, P.Z.; validation, P.Z., M.R., M.M. and F.F.; formal analysis, P.Z., M.R. and M.M.; investigation, P.Z.; resources, X.S.; writing—original draft preparation, P.Z. and M.R.; writing—review and editing, P.Z., M.R., M.M., F.F. and X.S.; visualization, P.Z. and M.R.; supervision, F.F. and X.S.; project administration, X.S.; funding acquisition, X.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data used for the experiments can be downloaded using the open-source code provided by the authors: https://github.com/pzinemanas/APNet.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	artificial intelligence
APNet	audio prototype network
ASC	automatic sound classification
CNN	convolutional neural network
CRNN	convolutional recurrent neural network
DNN	deep neural network
GDPR	general data protection regulation
GMM	Gaussian mixture model
LIME	local interpretable model-agnostic explanations
LSTM	long short-time memory
MFCC	mel frequency cepstral coefficients
MLP	multi-layer perceptron

References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 56–67. [CrossRef]
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 1 January 2021).
3. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*; Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
4. Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep Contextualized Word Representations. In *Volume 1 (Long Papers), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 2–7 February 2018*; Association for Computational Linguistics: New Orleans, LA, USA, 2018; pp. 2227–2237. [CrossRef]
5. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]

6. Schedl, M. Deep Learning in Music Recommendation Systems. *Front. Appl. Math. Stat.* **2019**, *5*, 44. [CrossRef]
7. Zhang, Q.; Wang, W.; Zhu, S.C. Examining CNN Representations With Respect to Dataset Bias. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; AAAI Press: New Orleans, LA, USA, 2018; pp. 4464–4473.
8. Chen, L.; Cruz, A.; Ramsey, S.; Dickson, C.J.; Duca, J.S.; Hornak, V.; Koes, D.R.; Kurtzman, T. Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening. *PLoS ONE* **2019**, *14*, 1–22. [CrossRef]
9. Menon, S.; Damian, A.; Hu, M.; Ravi, N.; Rudin, C. PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 2434–2442. [CrossRef]
10. Molnar, C. Interpretable Machine Learning 2019. Available online: https://christophm.github.io/interpretable-ml-book/ (accessed on 1 January 2021).
11. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef]
12. Xie, N.; Ras, G.; van Gerven, M.; Doran, D. Explainable Deep Learning: A Field Guide for the Uninitiated. *arXiv* **2020**, arXiv:2004.14545.
13. Krishnan, M. Against Interpretability: A Critical Examination of the Interpretability Problem in Machine Learning. *Philos. Technol.* **2020**, *33*, 487–502. [CrossRef]
14. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
15. Carlini, N.; Wagner, D. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In Proceedings of the IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; pp. 1–7. [CrossRef]
16. Qin, Y.; Carlini, N.; Cottrell, G.; Goodfellow, I.; Raffel, C. Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. In Proceedings of the 36th International Conference on Machine Learning (PMLR), Long Beach, CA, USA, 10–15 June 2019; Volume 97, pp. 5231–5240.
17. Goodman, B.; Flaxman, S. European Union Regulations on Algorithmic Decision-Making and a ”Right to Explanation”. *AI Mag.* **2017**, *38*, 50–57. [CrossRef]
18. Gilpin, L.H.; Bau, D.; Yuan, B.Z.; Bajwa, A.; Specter, M.; Kagal, L. Explaining Explanations: An Overview of Interpretability of Machine Learning. In Proceedings of the 5th IEEEInternational Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–4 October 2018; pp. 80–89. [CrossRef]
19. Zhang, Q.S.; Zhu, S.C. Visual interpretability for deep learning: A survey. *Front. Inf. Technol. Electron. Eng.* **2018**, *19*, 27–39. [CrossRef]
20. Lipton, Z. The Mythos of Model Interpretability. *Commun. ACM* **2016**, *61*. [CrossRef]
21. Li, O.; Liu, H.; Chen, C.; Rudin, C. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), New Orleans, LA, USA, 2–7 February 2018; Volume 32.
22. Alvarez Melis, D.; Jaakkola, T. Towards Robust Interpretability with Self-Explaining Neural Networks. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA,2018; Volume 31, pp. 7775–7784.
23. Hase, P.; Chen, C.; Li, O.; Rudin, C. Interpretable Image Recognition with Hierarchical Prototypes. In Proceedings of the The Seventh AAAI Conference on Human Computation and Crowdsourcing (HCOMP-19), Skamania Lodge, WA, USA, 28–30 October 2019; AAAI Press: Stevenson, WA, USA, 2019; Volume 7, pp. 32–40.
24. Chen, C.; Li, O.; Tao, D.; Barnett, A.J.; Su, J.; Rudin, C. This Looks Like That: Deep Learning for Interpretable Image Recognition. In *Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
25. Chen, Z.; Bei, Y.; Rudin, C. Concept whitening for interpretable image recognition. *Nat. Mach. Intell.* **2020**, *2*, 772–782. [CrossRef]
26. Guidotti, R.; Monreale, A.; Turini, F.; Pedreschi, D.; Giannotti, F. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.* **2018**, *51*. [CrossRef]
27. Ribeiro, M.T.; Singh, S.; Guestrin, C. "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), San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1135–1144. [CrossRef]
28. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27, pp. 3320–3328.
29. Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C.; Wexler, J.; Viegas, F.; Sayres, R. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *Machine Learning Research, Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018*; Dy, J., Krause, A., Eds.; PMLR: Stockholmsmässan, Sweden, 2018; Volume 80, pp. 2668–2677.

30. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
31. Adebayo, J.; Gilmer, J.; Muelly, M.; Goodfellow, I.; Hardt, M.; Kim, B. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31.
32. Siddharth, N.; Paige, B.; Van de Meent, J.W.; Desmaison, A.; Goodman, N.D.; Kohli, P.; Wood, F.; Torr, P.H. Learning disentangled representations with semi-supervised deep generative models. *arXiv* **2017**, arXiv:1706.00400.
33. Wiegreffe, S.; Pinter, Y. 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), Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Hong Kong, China, 2019; pp. 11–20.
34. Krstulović, S. Audio Event Recognition in the Smart Home. In *Computational Analysis of Sound Scenes and Events*; Virtanen, T., Plumbley, M.D., Ellis, D., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 335–371. [CrossRef]
35. Bello, J.P.; Mydlarz, C.; Salamon, J., Sound Analysis in Smart Cities. In *Computational Analysis of Sound Scenes and Events*; Virtanen, T., Plumbley, M.D., Ellis, D., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 373–397. [CrossRef]
36. Stowell, D., Computational Bioacoustic Scene Analysis. In *Computational Analysis of Sound Scenes and Events*; Virtanen, T., Plumbley, M.D., Ellis, D., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 303–333. [CrossRef]
37. Mesaros, A.; Heittola, T.; Virtanen, T. TUT database for acoustic scene classification and sound event detection. In Proceedings of the 24rd IEEE European Signal Processing Conference 2016, Budapest, Hungary, 29 August–2 September 2016. [CrossRef]
38. Kwan, C.; Ho, K.; Mei, G.; Li, Y.; Ren, Z.; Xu, R.; Zhang, Y.; Lao, D.; Stevenson, M.; Stanford, V.; Rochet, C. An Automated Acoustic System to Monitor and Classify Birds. *Eurasip J. Adv. Signal Process.* **2006**, *2006*, 1–19. [CrossRef]
39. Salamon, J.; Bello, J.P. Unsupervised feature learning for urban sound classification. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QSD, Australia, 19–24 April 2015; IEEE: New York, NY, USA, 2015; pp. 171–175. [CrossRef]
40. Cramer, J.; Wu, H.H.; Salamon, J.; Bello, J.P. Look, Listen and Learn More: Design Choices for Deep Audio Embeddings. In Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; IEEE: Brighton, UK, 2019; pp. 3852–3856.
41. Salamon, J.; Bello., J.P. Deep Convolutional Neural Networks and Data Augmentation For Environmental Sound Classification. *IEEE Signal Process. Lett.* **2017**, *24*, 279–283. [CrossRef]
42. Coimbra de Andrade, D.; Leo, S.; Loesener Da Silva Viana, M.; Bernkopf, C. A neural attention model for speech command recognition. *arXiv* **2018**, arXiv:1808.08929.
43. Cakir, E.; Parascandolo, G.; Heittola, T.; Huttunen, H.; Virtanen, T. Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. *Trans. Audio Speech Lang. Process. Spec. Issue Sound Scene Event Anal.* **2017**, *25*, 1291–1303. [CrossRef]
44. Muckenhirn, H.; Abrol, V.; Magimai-Doss, M.; Marcel, S. Understanding and Visualizing Raw Waveform-Based CNNs. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 2345–2349. [CrossRef]
45. Bach, S.; Binder, A.; Montavon, G.; Frederick Klauschen, K.R.M.; Samek, W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS ONE* **2015**, *10*. [CrossRef]
46. Montavon, G.; Lapuschkin, S.; Binder, A.; Samek, W.; Müller, K.R. Explaining nonlinear classification decisions with deep Taylor decomposition. *PAttern Recognit.* **2017**, *65*, 211–222. [CrossRef]
47. Becker, S.; Ackermann, M.; Lapuschkin, S.; Müller, K.R.; Samek, W. Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals. *arXiv* **2018**, arXiv:1807.03418.
48. Schiller, D.; Huber, T.; Lingenfelser, F.; Dietz, M.; Seiderer, A.; André, E. Relevance-Based Feature Masking: Improving Neural Network Based Whale Classification Through Explainable Artificial Intelligence. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 2423–2427. [CrossRef]
49. Mishra, S.; Sturm, B.L.; Dixon, S. Local Interpretable Model-Agnostic Explanations for Music Content Analysis. In Proceedings of the 18th International Society for Music Information Retrieval Conference ISMIR, Suzhou, China, 12–16 October 2020; pp. 537–543.
50. Mishra, S.; Benetos, E.; Sturm, B.L.T.; Dixon, S. Reliable Local Explanations for Machine Listening. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; IEEE: Glasgow, UK, 2020; pp. 1–8. [CrossRef]
51. Choi, K.; Fazekas, G.; Sandler, M.; Cho, K. Transfer learning for music classification and regression tasks. In Proceedings of the 18th International Society for Music Information Retrieval Conference ISMIR, Suzhou, China, 23–27 October 2017.
52. Li, C.; Yuan, P.; Lee, H. What Does a Network Layer Hear? Analyzing Hidden Representations of End-to-End ASR Through Speech Synthesis. In Proceedings of the CASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 6434–6438.
53. Thickstun, J.; Harchaoui, Z.; Kakade, S.M. Learning Features of Music from Scratch. In Proceedings of the International Conference on Learning Representations ICLR, Toulon, France, 24–26 April 2017.
54. Lee, J.; Park, J.; Kim, K.L.; Nam, J. SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification. *Appl. Sci.* **2018**, *8*, 150. [CrossRef]

55. Tax, T.M.S.; Antich, J.L.D.; Purwins, H.; Maaløe, L. Utilizing Domain Knowledge in End-to-End Audio Processing. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
56. Loweimi, E.; Bell, P.; Renals, S. On Learning Interpretable CNNs with Parametric Modulated Kernel-Based Filters. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 3480–3484. [CrossRef]
57. Pan, Y.; Mirheidari, B.; Tu, Z.; O'Malley, R.; Walker, T.; Venneri, A.; Reuber, M.; Blackburn, D.; Christensen, H. Acoustic Feature Extraction with Interpretable Deep Neural Network for Neurodegenerative Related Disorder Classification. In Proceedings of Interspeech 2020, Shanghai, China, 25–29 October 2020; pp. 4806–4810. [CrossRef]
58. Won, M.; Chun, S.; Nieto, O.; Serrc, X. Data-Driven Harmonic Filters for Audio Representation Learning. In Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–9 May 2020; pp. 536–540. [CrossRef]
59. Cakır, E.; Virtanen, T. End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input. In Proceedings of the 2018 IEEE International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; [CrossRef]
60. Zinemanas, P.; Cancela, P.; Rocamora, M. End-to-end Convolutional Neural Networks for Sound Event Detection in Urban Environments. In Proceedings of the 24th IEEE Conference of Open Innovations Association FRUCT, Moscow, Russia, 8–12 April 2019. [CrossRef]
61. Chaki, S.; Doshi, P.; Bhattacharya, S.; Patnaik, P. Explaining perceived emotion predictions in music: An attentive approach. In Proceedings of the 21st International Society for Music Information Retrieval Conference ISMIR, Montréal, QC, Canada, 10–16 October 2020; pp. 150–156.
62. Jalal, M.A.; Milner, R.; Hain, T. Empirical Interpretation of Speech Emotion Perception with Attention Based Model for Speech Emotion Recognition. In Proceedings of Interspeech 2020, Shanghai, China, 25–29 October 2020; pp. 4113–4117. [CrossRef]
63. Won, M.; Chun, S.; Serra, X. Toward Interpretable Music Tagging with Self-Attention. *arXiv* **2019**, arXiv:1906.04972.
64. Yang, S.; Liu, A.T.; Lee, H. Understanding Self-Attention of Self-Supervised Audio Transformers. In Proceedings of Interspeech 2020, Shanghai, China, 25–29 October 2020; pp. 3785–3789. [CrossRef]
65. Luo, Y.J.; Cheuk, K.W.; Nakano, T.; Goto, M.; Herremans, D. Unsupervised Disentanglement of Pitch and Timbre for Isolated Musical Instrument Sounds. In Proceedings of the 21st International Society for Music Information Retrieval Conference ISMIR, Montréal, QC, Canada, 10–16 October 2020.
66. Wang, Z.; Wang, D.; Zhang, Y.; Xia, G. Learning Interpretable Representation for Controllable Polyphonic Music Generation. In Proceedings of the 21st International Society for Music Information Retrieval Conference ISMIR, Montréal, QC, Canada, 10–16 October 2020; pp. 662–669.
67. Chowdhury, S.; Vall, A.; Haunschmid, V.; Widmer, G. Towards Explainable Music Emotion Recognition: The Route via Mid-level Features. In Proceedings of the 20th International Society for Music Information Retrieval Conference ISMIR, Delft, The Netherlands, 4–8 November 2019; pp. 237–243.
68. De Berardinis, J.; Cangelosi, A.; Coutinho, E. 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 ISMIR, Montréal, QC, Canada, 10–16 October 2020.
69. Kelz, R.; Widmer, G. Towards Interpretable Polyphonic Transcription with Invertible Neural Networks. In Proceedings of the 20th International Society for Music Information Retrieval Conference ISMIR, Delft, The Netherlands, 4–8 November 2019; pp. 376–383.
70. Ardizzone, L.; Kruse, J.; Rother, C.; Köthe, U. Analyzing Inverse Problems with Invertible Neural Networks. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
71. Agrawal, P.; Ganapathy, S. Interpretable Representation Learning for Speech and Audio Signals Based on Relevance Weighting. *IEEE Acm Trans. Audio Speech Lang. Process.* **2020**, *28*, 2823–2836. [CrossRef]
72. Luo, J.H.; Wu, J.; Lin, W. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5058–5066.
73. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
74. Maas, A.L.; Hannun, A.Y.; Ndrew Y. N. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In Proceedings of the 30th International Conference on Machine Learning JMLR, Atlanta, GA, USA, 16–21 June 2013; Volume 28.
75. Pons, J.; Lidy, T.; Serra, X. Experimenting with musically motivated convolutional neural networks. In Proceedings of the 2016 IEEE 14th International Workshop on Content-Based Multimedia Indexing (CBMI), Bucharest, Romania, 15–17 June 2016; pp. 1–6. [CrossRef]
76. Salamon, J.; Jacoby, C.; Bello, J.P. A Dataset and Taxonomy for Urban Sound Research. In Proceedings of the 22nd ACM International Conference on Multimedia (MM '14), New York, NY, USA, 3–7 November 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1041–1044. [CrossRef]
77. Font, F.; Roma, G.; Serra, X. Freesound Technical Demo. In Proceedings of the ACM International Conference on Multimedia (MM'13), Barcelona, Spain, 21–25 October 2013; ACM: Barcelona, Spain, 2013; pp. 411–412. [CrossRef]
78. Lostanlen, V.; Cella, C.E. Deep convolutional networks on the pitch spiral for musical instrument recognition. In Proceedings of the 17th International Society for Music Information Retrieval Conference ISMIR, New York, NY, USA, 7–11 August 2016.

79. Bittner, R.; Salamon, J.; Tierney, M.; Mauch, M.; Cannam, C.; Bello, J. MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In Proceedings of the 15th International Society for Music Information Retrieval Conference ISMIR, Taipei, Taiwan, 27–31 October 2014.
80. Joder, C.; Essid, S.; Richard, G. Temporal Integration for Audio Classification With Application to Musical Instrument Classification. *IEEE Trans. Audio Speech Lang. Process.* **2009**, *17*, 174–186. [CrossRef]
81. Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv* **2018**, arXiv:1804.03209.
82. 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. In Proceedings of the Fifth Workshop on Detection and Classification of Acoustic Scenes and Events DCASE, Tokyo, Japan, 2–3 November 2020; pp. 240–244.
83. Griffin, D.; Lim, J. Signal estimation from modified short-time Fourier transform. *IEEE Trans. Acoust. Speech Signal Process.* **1984**, *32*, 236–243. [CrossRef]
84. McFee, B.; McVicar, M.; Raffel, C.; Liang, D.; Nieto, O.; Battenberg, E.; Moore, J.; Ellis, D.; Yamamoto, R.; Bittner, R.; et al. librosa: 0.4.1. *Zenodo* **2015**. [CrossRef]