Listen to Interpret: Post-hoc Interpretability for Audio Networks with NMF

Jayneel Parekh[1] **Sanjeel Parekh**[1,2*] **Pavlo Mozharovskyi**[1]
Florence d'Alché-Buc[1] **Gaël Richard**[1]
[1]LTCI, Télécom Paris, Institut Polytechnique de Paris [2]Audio Analytic
{jayneel.parekh,pavlo.mozharovskyi,florence.dalche,gael.richard}@telecom-paris.fr
sanjeel.parekh@audioanalytic.com

Abstract

This paper tackles *post-hoc* interpretability for audio processing networks. Our goal is to interpret decisions of a trained network in terms of high-level audio objects that are also listenable for the end-user. To this end, we propose a novel interpreter design that incorporates non-negative matrix factorization (NMF). In particular, a regularized interpreter module is trained to take hidden layer representations of the targeted network as input and produce time activations of pre-learnt NMF components as intermediate outputs. Our methodology allows us to generate intuitive audio-based interpretations that explicitly enhance parts of the input signal most relevant for a network's decision. We demonstrate our method's applicability on popular benchmarks, including a real-world multi-label classification task.

1 Introduction

Deep learning models, while state-of-the-art for several tasks in domains such as computer vision, natural language processing and audio, are typically not interpretable. Their increasing use, especially in decision-critical domains, necessitates interpreting their decisions. A good interpretation is often characterized by its understandability for the end users (see for instance [16]). More importantly, attributes that aid understandability may largely be dependent on the data modality. In this paper, our aim is to generate *post-hoc* human–understandable interpretations for deep networks that process the audio modality. Here, *post-hoc* interpretability refers to the problem of interpreting decisions of a fixed pre–trained network.

Traditional approaches generate interpretations through input attribution, either directly on the raw input features or on a given simplified representation [33, 43, 39, 29]. To generate more understandable interpretations, a small number of approaches consider other means, such as logical rules [40], sentences [19] and high-level concepts [15].

Most existing *post-hoc* interpretability methods are primarily designed for application to images and tabular data. This limits their applicability to other data modalities such as audio. Although many audio processing networks operate on spectrogram-like representations, which can be seen as 2D time-frequency images, a visualization or attribution in this space is not as meaningful to a common user as it is for images [27].

This leads us to build an interpretation system that takes into account audio–specific understandability features. We motivate these features through an example: suppose an audio event detection network deployed in a house recognizes an "alarm" sound event. An ideal interpreter for this classification decision would have the ability to "show" that it was indeed an alarm sound that triggered this decision. To do so, it must be able to localize the alarm amid other events in the house (for *e.g.* dog

*Work conducted while the author was at Télécom Paris

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

barks, baby cries, background noise *etc.*) and make it listenable for the end–user. It is important to highlight here the role of listenable interpretations for better understanding of an audio network's decisions – note that it would be much less meaningful for a human to see the alarm sound as highlighted parts of a spectrogram. Thus making the following aspects important for our system design: (i) generating interpretations in terms of high–level audio objects that constitute a scene, (ii) segmenting parts of the input signal most relevant for a decision and providing it as listenable audio. It's worth emphasizing that audio interpretability is not the same as classical tasks of separation or denoising. These tasks involve recovering complete object of interest in the output audio. On the other hand, a classifier network might focus more on salient regions. When interpreting its decision and making it listenable we expect to uncover such regions and not necessarily the complete object of interest.

To this end, we propose a novel *post–hoc* interpreter for audio that employs a popular signal decomposition technique called Non–negative Matrix Factorization (NMF; Lee and Seung [26]). NMF seeks to decompose an audio signal into constituent spectral patterns and their temporal activations. Unlike principal component analysis, NMF is known to provide part–based decompositions [12]. Owing to these properties, we first use NMF to pre-learn a spectral pattern dictionary on our training data. This dictionary is then incorporated as a fixed decoder within our interpretation module. Specifically, we train our system to determine an intermediate encoding that performs two roles: (i) is able to reconstruct the input through the fixed NMF dictionary decoder, thus corresponding to time activations for dictionary components, (ii) at the same time, a function of this encoding is able to mimic the classifier's output. Training with these constraints allows us to generate, for any classifier decision, importance values over spectral patterns in our dictionary. Listenable interpretations are readily produced by inverting most important NMF spectral patterns back to the time domain.

In summary, we make the following contributions:

- We propose a novel NMF-based interpreter module for *post-hoc* interpretability that generates interpretations in terms of meaningful high–level audio objects, listenable for the end–user.
- We present an original formulation that constrains the interpreter encoding through two loss functions, one for input reconstruction through NMF dictionary and the other for fidelity to the network's decision. From a learning perspective, we show a new way to link NMF with deep neural networks, especially for generating interpretations.
- We extensively evaluate on two popular audio event analysis benchmarks, tackling both multi–class and multi–label classification tasks. The dataset for the latter is very challenging due to its collection in noisy real–world setting. Our method's design allows us to simulate feature removal and perform *faithfulness* evaluation.

2 Related Works

In this section, we position our work in relation to: (i) interpretability methods for audio, (ii) methods for concept–based interpretability and, (iii) use of NMF within the audio community, in particular, attempts to link it with deep networks.

Interpretability methods for audio Some approaches [5, 50] have shown usability of attention/visualization techniques for interpreting audio processing networks or generated instance-wise feature importance [52] for time-series data [47]. However, our focus is on methods that attempt to address audio interpretability beyond image-based visualizations or raw input attribution. Muckenhirn et al. [34] illustrate usefulness of GuidedBackprop [45] for analyzing CNNs operating on raw 1D waveforms by analyzing relevance signal in frequency domain. This analysis does not extend to spectrogram input or address listenability of interpretations. A few works have applied the popular LIME algorithm with a simplified input representation more suited for audio. In particular, SLIME [31, 32] proposes to segment the input along time or frequency. The input is perturbed by switching "on/off" the individual segments. AudioLIME [18, 10] proposes to separate input using predefined sources to create the simplified representation. AudioLIME arguably generates more meaningful interpretations than SLIME as it relies on audio objects readily listenable for end-user. However, it can only be applied for limited applications as it requires existence of known and meaningful predefined sources that compose the input audio. APNet [53] takes another promising direction by

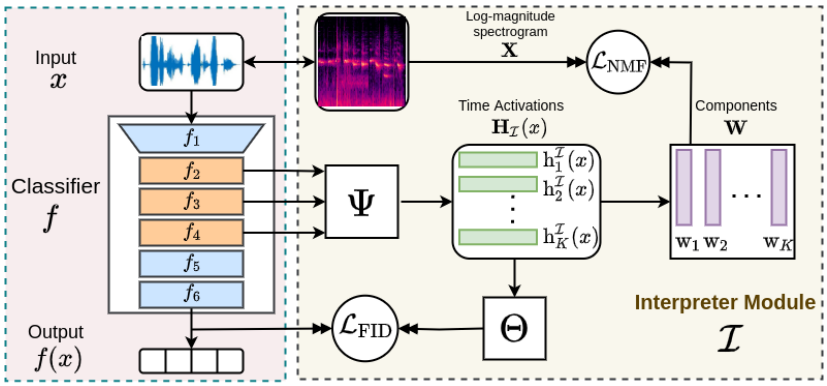


Figure 1: **System overview**: The interpretation module (right block) accesses hidden layer outputs of the network being interpreted (left block). These are used to predict an intermediate encoding. Through regularization terms, we encourage this encoding to both mimic the classifier's output and also serve as the time activations of a pre-learnt NMF dictionary.

extending interpretable prototypical networks for audio input. However, they propose an interpretable system *by-design*. They don't tackle the problem of *post-hoc* interpretation.

Concept-based interpretability Our method relies on high-level objects for interpretation. In this sense, it is most closely related to *post–hoc* concept-based methods [21, 15]. An interesting approach is that of *post-hoc* version of FLINT [35] with whom we share the idea of utilizing the hidden layers and loss functions to encourage interpretability. However we crucially differ from FLINT and other related approaches in concept representation and their applicability for audio interpretations. FLINT represents concepts by a dictionary of attribute functions over input space. The learnt concepts are not obviously comprehensible to a user, requiring a separate visualization pipeline to get insights. Approaches based on TCAV [21], such as ACE [15], ConceptSHAP [51], define concept using a set of images and learn a representation for it in terms of hidden layers of the network, termed as concept activation vector (CAV). These designs for concepts are not related to our NMF-inspired dictionary representation. Importantly, none of the above mentioned approaches can generate listenable interpretations which is key for understandability of audio processing networks.

NMF for audio has been widely used for numerous tasks ranging from separation to transcription [44, 48, 11, 6]. Its traditional usage as a supervised dictionary or feature learning method involves learning class-wise dictionaries over training data [12]. Time activations, which are the so-called features, are generated for any data point by projecting it onto the learnt dictionaries. Extracted features can subsequently be used for downstream tasks such as classification. Bisot et al. [7] couple NMF-based features with neural networks to boost performance of acoustic scene classification.

NMF has also been successfully employed with audio–visual deep learning models for separation [13] and classification [36]. Another line of research explored unfolding iterations of different NMF optimization algorithms as a deep neural network [24, 49]. These systems, commonly known as Deep NMF, have primarily been used for audio source separation tasks.

While these works share with us the idea of combining neural networks and NMF, there is no overlap between our goals and methodologies. Unlike aforementioned studies, we wish to investigate a classifier's decision in a post-hoc manner using NMF as a regularizer. Furthermore, to our best knowledge, attempting to regress temporal activations of a fixed NMF dictionary by accessing intermediate layers of an audio classification network is novel even within the NMF literature.

3 System Design

We begin this section with a brief note on data notation and NMF. Subsequently, in Sec. 3.1, we discuss interpreter module's design and learning. This is followed by a description of our interpretation generation methodology in Sec. 3.2. An overview of the proposed system is presented in Fig. 1.

Data notation. We denote a training dataset by $\mathcal{S} := (x, y)_{i=1}^{N}$, where x is the time domain audio signal and y, a label vector. The label vector could be a one-hot or binary encoding depending

upon a multi-class or multi-label dataset, respectively. Since multiple audio representations are used in this paper, a note on their notation is in order. Very often, audio signals are processed in the frequency domain through a short-time Fourier transform (STFT) on x called spectrogram. Log-mel spectrograms are a popular input to audio classification networks [37], which is also the one we use in this paper. To keep notation simple, we refer to input of the network with x. For NMF however, we favor a representation of x that can be easily inverted back to the time-domain and use a log–magnitude spectrogram $\mathbf{X}$ that is computed by applying an element-wise transformation $x_0 \rightarrow \log(1 + x_0)$ on the magnitude spectrogram. This is preferred over using magnitude spectrograms as it corresponds more closely to human perception of sound intensity [17].

NMF basics. NMF is a popular technique for unsupervised decomposition of audio signals [3]. Given any positive time–frequency representation $\mathbf{X} \in \mathbb{R}_+^{F \times T}$ consisting of F frequency bins and T time frames, NMF decomposes it into a product of two non-negative matrices, such that,

$$\mathbf{X} \approx \mathbf{W}\mathbf{H}$$

Here, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K] \in \mathbb{R}_+^{F \times K}$ is interpreted as the spectral pattern or dictionary matrix containing K components and $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_K]^\intercal \in \mathbb{R}_+^{K \times T}$ a matrix containing the corresponding time activations. Typically, a β-divergence measure between $\mathbf{X}$ and $\mathbf{W}\mathbf{H}$ is minimized and multiplicative updates are used for estimating $\mathbf{W}$ and $\mathbf{H}$ [26]. Note that it is possible to reconstruct signal corresponding to each or a group of spectral components. This is typically done using a procedure called soft–masking. For a single component k, this is written as,

$$\mathbf{X}_k = \frac{\mathbf{w}_k \mathbf{h}_k{}^\intercal}{\mathbf{W}\mathbf{H}} \odot \mathbf{X}$$

Both $./.$ and $\odot$ are element-wise operations. If $\mathbf{X}$ is an invertible representation of the magnitude spectrogram, time domain signal for $\mathbf{X}_k$ is easily recovered using the inverse STFT operation. We extensively utilize this procedure for generating listenable interpretations. NMF can also be used for dictionary learning, by estimating $\mathbf{W}$ on a training dataset matrix $\mathbf{X}_{\text{train}}$. As discussed later, we use a variant of NMF called Sparse-NMF [25] to pre-learn dictionary for subsequent usage in the interpretation module.

3.1 Interpreter Design

As depicted in Fig. 1 the interpreter module $\mathcal{I}$ contains two components: an interpreter network and a NMF dictionary decoder. The so-called interpreter network computes the following function $x \mapsto \Theta \circ \Psi \circ f_{\mathcal{I}}(x)$ where Ψ is the function responsible for generating an intermediate encoding from hidden layer representations of the classification network, and Θ attempts to mimic the classifier's output given the intermediate representation. The NMF decoder based on a pre-trained dictionary plays two roles: (i) during training, it constrains the intermediate representation to correspond to time activations of a pre-learnt spectral pattern dictionary and (ii) when interpreting a classifier's prediction, it is used to build listenable interpretation. To the best of our knowledge, this is an original usage of NMF that allows us to interpret a network's decisions in terms of a fixed dictionary.

Design of Ψ. The function Ψ processes outputs of a set of hidden layers of the classifier, given by $f_{\mathcal{I}}(x)$. It's output, $\Psi(f_{\mathcal{I}}(x)) \in \mathbb{R}_+^{K \times T}$ produces an intermediate encoding of the interpreter. For simplicity, we will denote this intermediate encoding as $\mathbf{H}_{\mathcal{I}}(x) = \Psi \circ f_{\mathcal{I}}(x)$, a function over input x.

In practice, Ψ is modelled as a neural network that takes as input convolutional feature maps from different layers of f. To concatenate and perform joint processing on them, each feature map is first appropriately transformed to ensure same width and height dimensions. Two important aspects were kept in mind while designing subsequent layers of Ψ. Firstly, audio feature maps for spectrogram-like inputs naturally contain the notion of time and frequency along the width and height dimensions. Secondly, through appropriate regularization we wish to produce an intermediate encoding that also serves as time activations of the pre-learnt NMF dictionary, a matrix of dimensions $K \times T$. To achieve this, we continuously downsample on the frequency axis and upsample the time axis to T frames. Similarly, the number of input feature maps is re-sampled to reach a size of K, equal to the number of components in dictionary $\mathbf{W}$. Additionally, we ensure non-negativity of $\mathbf{H}_{\mathcal{I}}(x)$ through use of ReLU activation. All learsnable parameters of Ψ are denoted by V_Ψ.

Design of Θ. $\mathbf{H}_{\mathcal{I}}(x)$, the intermediate encoding output by Ψ is then fed to Θ, which aims to mimic output of the classifier. This directly helps in learning a representation which can interpret

$f(x)$. Its design consists of two parts. The first part pools activations $\mathbf{H}_{\mathcal{I}}(x)$ across time. While this pooling can be implemented in multiple ways, we opt for attention–based pooling [20], *i.e.*, $\mathbf{z} = \sum_{t=1}^{T} \mathbf{H}_{\mathcal{I}}(x)\mathbf{a}$, where $\mathbf{a} \in \mathbb{R}^T$ are the attention weights and $\mathbf{z} \in \mathbb{R}^K$ is the pooled vector. The pooled representation vector is passed through a linear layer. This is followed by an appropriate activation function to convert its output to probabilities, that is, softmax for multi-class classification and sigmoïd for multi-label classification. All learnable parameters of Θ are denoted by V_Θ.

Fidelity loss. Generalized cross–entropy between interpreter's output $\Theta(\mathbf{H}_{\mathcal{I}}(x))$ and classifier's output $f(x)$ is minimized to encourage interpreter to mimic the classifier. For multi-class classification this loss function is written as,

$$\mathcal{L}_{\text{FID}}(x, V_\Psi, V_\Theta) = -f(x)^\intercal \log(\Theta(\mathbf{H}_{\mathcal{I}}(x))) \tag{1}$$

On the other hand, for multi-label classification this loss reads,

$$\mathcal{L}_{\text{FID}}(x, V_\Psi, V_\Theta) = -\sum f(x) \odot \log(\Theta(\mathbf{H}_{\mathcal{I}}(x))) + (1 - f(x)) \odot \log(1 - \Theta(\mathbf{H}_{\mathcal{I}}(x))). \tag{2}$$

Here $\odot$ denotes element-wise multiplication.

NMF dictionary decoder and regularization. We additionally constrain the intermediate encoding, such that, when fed to a decoder it is able to reconstruct the input audio. As already discussed, we choose this decoder to be a pre-learnt NMF dictionary, $\mathbf{W}$. Formally, through $\mathcal{L}_{\text{NMF}}$ we require $\mathbf{H}_{\mathcal{I}}(x)$ to approximate log-magnitude spectrogram $\mathbf{X}$ of input audio x as $\mathbf{X} \approx \mathbf{W}\mathbf{H}_{\mathcal{I}}(x)$:

$$\mathcal{L}_{\text{NMF}}(x, V_\Psi) = \|\mathbf{X} - \mathbf{W}\mathbf{H}_{\mathcal{I}}(x)\|_2^2. \tag{3}$$

This allows us to consider $\mathbf{H}_{\mathcal{I}}(x)$ as a time activation matrix for $\mathbf{W}$.

Training loss. In addition to $\mathcal{L}_{\text{FID}}$ and $\mathcal{L}_{\text{NMF}}$, we impose ℓ_1 regularization on $\mathbf{H}_{\mathcal{I}}(x)$ to encourage sparsity and well-behavedness, as is often done in classical NMF [25]. The complete training loss function over our training dataset $\mathcal{S}$ can thus be given as:

$$\mathcal{L}(V_\Psi, V_\Theta) = \sum_{x \in \mathcal{S}} \mathcal{L}_{\text{FID}}(x, V_\Psi, V_\Theta) + \alpha \mathcal{L}_{\text{NMF}}(x, V_\Psi) + \beta ||\mathbf{H}_{\mathcal{I}}(x)||_1 \tag{4}$$

where $\alpha, \beta \geq 0$ are loss hyperparameters. All the parameters of the system are constituted in the functions Ψ, Θ and dictionary $\mathbf{W}$. Since $\mathbf{W}$ is pre-learnt and fixed, the training loss $\mathcal{L}$ is optimized only w.r.t V_Ψ, V_Θ. As a reminder, when training the interpreter for post-hoc analysis, the classifier network is kept fixed.

Algorithm 1 Learning algorithm

1: **Input:** Classifier f, Training data $\mathcal{S}$, parameters $V = \{V_\Psi, V_\Theta\}$, hyperparameters $\{\alpha, \beta, \mu\}$, number of batches B, number of training epochs N_{epoch}
2: $\mathbf{W} \leftarrow$ PRE-LEARN NMF DICTIONARY $(\mathcal{S}, \mu)$ {// Sparse-NMF algorithm}
3: Random initialization of parameters V_0
4: $\hat{V} \leftarrow$ TRAIN $(f, \mathcal{S}, \mathbf{W}, V_0, \alpha, \beta, B, N_{\text{epoch}})$ {// Train with $\mathcal{L}$ in Eq. 4}
5: **Output**: $\hat{V} = \{\hat{V}_\Psi, \hat{V}_\Theta\}$

Learning algorithm. The complete learning pipeline is presented in Algorithm 1. The learnable parameters of the interpreter module are given by $V = \{V_\Psi, V_\Theta\}$. The pre-specified dictionary (Step 2 in Algorithm 1) is learnt using Sparse-NMF [25]. Specifically, the following optimization problem is solved through multiplicative updates to pre-learn $\mathbf{W}$:

$$\min D(\mathbf{X}_{\text{train}}|\mathbf{W}\mathbf{H}) + \mu\|\mathbf{H}\|_1 \quad s.t. \mathbf{W} \geq 0, \mathbf{H} \geq 0, \|\mathbf{w}_k\| = 1, \ \forall k. \tag{5}$$

Here $D(.|.)$ is a divergence cost function. In practice, euclidean distance is used. The reader is referred to appendix A.1 for more details regarding Sparse-NMF optimization problem, such as construction of $\mathbf{X}_{\text{train}}$ on our datasets.

3.2 Interpretation generation

Finally, to generate audio that interprets the classifier's decision for a sample x and a predicted class c, we follow a two-step procedure: The first step consists of selecting the components which are

considered "important" for the prediction. This is determined by estimating their relevance using the pooled time activations in Θ and the weights for linear layer, and then thresholding it. Precisely, given a sample x, the pooled activations are computed as $\mathbf{z} = \mathbf{H}_{\mathcal{I}}(x)\mathbf{a}$. Denoting the weights for class c in the linear layer as θ_c^w, the relevance of component k is estimated as $r_{k,c,x} = \frac{(\mathbf{z}_k \theta_{c,k}^w)}{\max_l |\mathbf{z}_l \theta_{c,l}^w|}$. This is essentially the normalized contribution of component k in the output logit for class c. Given a threshold τ, the selected set of components are computed as $L_{c,x} = \{k : r_{k,c,x} > \tau\}$.

The second step consists of estimating a time domain signal for each relevant component $k \in L_{c,x}$ and also for set $L_{c,x}$ as a whole. In this paper, we refer to the latter as the generated interpretation audio, x_{int}. For certain classes, it may also be meaningful to listen to each individual component, x_k. As discussed earlier under NMF basics, estimating time domain signals from spectral patterns and their activations typically involves a soft–masking and inverse STFT procedure. The inversion is performed using input audio phase $\mathbf{P}_x$. We detail this step with appropriate equations in Algorithm 2.

Algorithm 2 Audio interpretation generation

Input: log-magnitude spectrogram $\mathbf{X}$, input phase $\mathbf{P}_x$ components $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$, time activations $\mathbf{H}_{\mathcal{I}}(x) = [\mathbf{h}_1^{\mathcal{I}}(x), \ldots, \mathbf{h}_K^{\mathcal{I}}(x)]^\intercal$, set of selected components $L_{c,x} = \{k_1, \ldots, k_B\}$.
for all $k \in L_{c,x}$ **do**
  $\mathbf{X}_k \leftarrow \frac{\mathbf{w}_k \mathbf{h}_k^{\mathcal{I}}(x)^\intercal}{\sum_{l=1}^{K} \mathbf{w}_l \mathbf{h}_l^{\mathcal{I}}(x)^\intercal} \odot \mathbf{X}$ {// Soft masking}
  $x_k = \text{INV}(\mathbf{X}_k, \mathbf{P}_x)$ {// Inverse STFT}
end for
$\mathbf{X}_{\text{int}} \leftarrow \sum_{k \in L_{c,x}} \mathbf{X}_k$
$x_{\text{int}} = \text{INV}(\mathbf{X}_{\text{int}}, \mathbf{P}_x)$
Output: $\{x_{k_1}, \ldots, x_{k_B}\}$, x_{int}

4 Experiments

We experiment with two popular audio event analysis benchmarks, namely ESC-50 [38] and SONYC-UST [9]. While the former is a multiclass environmental sound classification dataset, the latter appeared for DCASE 2019 and 2020 multi-label urban sound tagging task. We quantitatively and qualitatively evaluate different aspects of our interpretations, including a subjective evaluation carried out on SONYC-UST. The implementation of our system is available on GitHub[2] This section is organized as follows: quantitative metrics and baselines are discussed in Sec. 4.1 followed by implementation details in Sec. 4.2. Experiments on ESC-50 and SONYC-UST are detailed in Sec. 4.3 and Sec. 4.4, respectively. We discuss some limitations in Sec. 4.5.

4.1 Quantitative metrics and baselines

Metrics. We quantitatively evaluate our interpretations in two ways. First, by evaluating how well it agrees with the classifier's output. For multi-class classification, this is done by computing fraction of samples where the class predicted by f is among the top-k classes predicted by the interpreter. We refer to this as the *top-k fidelity*. To compute *fidelity* on multi-label classification tasks, we compute Area Under Precision-Recall Curve (AUPRC) based metrics between the classifier output $f(x)$ and its approximation by interpreter $\Theta(\mathbf{H}_{\mathcal{I}}(x))$. We compute macro-AUPRC, micro-AUPRC. Additionally, we report the maximum micro F1-score over different thresholds for the interpreter's output.

We also conduct a *faithfulness* evaluation for our interpretations. In general for any interpretability method, *faithfulness* tries to assess if the features identified to be of high relevance are *truly* important in classifier's prediction [1]. Since a "ground-truth" importance measure for features is rarely available, attribution based methods evaluate faithfulness by performing feature removal (generally by setting feature value to 0) and observing the change in classifier's output [1]. However, it is hard to conduct such evaluation for non-attribution or concept based interpretation methods on data modalities like image/audio, as simulating feature removal from input is not evident in these cases.

[2] `https://github.com/jayneelparekh/L2I-code`

Interestingly, our interpretation module design allows us to simulate removal of a set of components from the input. Given any sample x with predicted class c, we remove the set of relevant components $L_{c,x} = \{k : r_{k,c,x} > \tau\}$ by creating a new time domain signal $x_2 = \text{INV}(\mathbf{X}_2, \mathbf{P}_x)$, where $\mathbf{X}_2 = \mathbf{X} - \sum_{l \in L_{c,x}} \mathbf{X}_l$. We define faithfulness of the interpretation to classifier f for sample x with:

$$\text{FF}_x = f(x)_c - f(x_2)_c \tag{6}$$

where $f(x)_c, f(x_2)_c$ denote the output probabilities for class c. It should be noted that this strategy to simulate removal may introduce artifacts in the input that can affect the classifier's output unpredictably. Also, interpretations on samples with poor fidelity can lead to negative FF_x. Both of these observations point to the potential instability and outlying values for this metric. Thus, we report the final faithfulness of the system as median of FF_x over test set, denoted by $\text{FF}_{\text{median}}$. A positive $\text{FF}_{\text{median}}$ would signify that interpretations generally tend to be faithful to the classifier.

Evaluated systems. We denote our proposed Listen to Interpret (L2I) system, with attention based pooling in Θ by L2I + Θ_{ATT}. The most suitable baselines to benchmark its fidelity are *post-hoc* methods that approximate the classifier over input space with a single surrogate model. We select two state-of-the-art systems, FLINT [35] and VIBI [4]. A variant of our own proposed method, L2I + Θ_{MAX}, is also evaluated. Herein, attention is replaced with 1D max-pooling operation. Implementation details of the baselines are discussed in appendix A.7.

Faithfulness benchmarking: As already discussed, it is not possible to measure faithfulness for concept-based *post-hoc* interpretability approaches. While measurement for input attribution based approaches is possible, the interpretations themselves and the feature removal strategies are different, making comparisons with our system significantly less meaningful. We thus compare our faithfulness against a *Random Baseline*, wherein randomly chosen components are removed. To compare fairly, we remove the same number of components that are present in $L_{c,x}$ on average. This would validate that, if the interpreter selects *truly* important components for the classifier's decision, then randomly removing the less important ones should not cause a drop in the predicted class probability.

We also emphasize at this point that works related to audio interpretability (see Sec. 2), are not suitable for comparison on these metrics. Particularly, APNet [53] is not designed for *post-hoc* interpretations. AudioLIME [18] is not applicable on our tasks as it requires known predefined audio sources. Moreover, SLIME [32] and AudioLIME still rely on LIME [39] for interpretations. It is a feature-attribution method that approximates a classifier for *each* sample separately. As discussed before, these characteristics are not suitable for comparison on our metrics.

4.2 Implementation details

Classification network. We interpret a VGG-style convolutional neural network proposed by Kumar et al. [23]. This network was chosen due to its popularity and applicability for various audio scene and event classification tasks. It can process variable length audio and has been pretrained on AudioSet [14], a large-scale weakly labeled dataset for sound events. Further details about the network and its fine-tuning can be found in appendix A.2.

Hyperparameters and training. The hidden layers input to the interpreter module are selected from the convolutional block outputs. As is often the case with CNNs, the latter layers are expected to capture higher-order features. We thus select the last three convolutional block outputs as input to the network Ψ. For ESC-50, we pre-learn the dictionary $\mathbf{W}$ with $K = 100$ components and for SONYC-UST, we learn with $K = 80$ components. Reasons for choice of K are discussed in appendix A.3. Ablation studies for other hyperparameters are in appendix A.4.

4.3 Experiments on ESC-50

Dataset. ESC-50 [38] is a popular benchmark for environmental sound classification task. It is a multi-class dataset that contains 2000 audio recordings of 50 different environmental sounds. The classes are broadly arranged in five categories namely, animals, natural soundscapes/water sounds, human/non-speech sounds, interior/domestic sounds, exterior/urban noises. Each clip is five-seconds long and has been extracted from publicly available recordings on the `freesound.org` project. The dataset is prearranged into 5 folds.

Classifier performance. The classifier achieves an accuracy of $82.5 \pm 1.9\%$ over the 5 folds, higher than the average human accuracy of 81.3% on ESC-50.

System	Fidelity (in %)		
	top-1	top-3	top-5
L2I + Θ_{ATT}	65.7 ± 2.8	81.8 ± 2.2	88.2 ± 1.7
L2I + Θ_{MAX}	73.3 ± 2.3	87.8 ± 1.8	92.7 ± 1.2
FLINT [35]	73.5 ± 2.3	89.1 ± 0.4	93.4 ± 0.9
VIBI [4]	27.7 ± 2.3	45.4 ± 2.2	53.0 ± 1.8

System	Threshold τ	$\text{FF}_{\text{median}}$
L2I + Θ_{ATT}	$\tau = 0.9$	0.002
	$\tau = 0.7$	0.004
	$\tau = 0.5$	0.012
	$\tau = 0.3$	0.040
	$\tau = 0.1$	0.113
Random Baseline	$\tau = 0.1$	$< 10^{-4}$

Table 1: Quantitative results on ESC-50 environmental sound classification test data. (Left) top-k fidelity (in %). FLINT and VIBI help benchmark fidelity but are not themselves suitable for audio interpretations. (Right) Faithfulness results (drop in probability) for different thresholds, τ. We report $\text{FF}_{\text{median}}$ for proposed L2I + Θ_{ATT} and the Random Baseline.

Quantitative results. Mean and standard deviation of top-k fidelity is calculated over the 5 folds. We show these results in Table 1 (Left) for $k = 1, 3, 5$. Among the four systems, VIBI performs the worst in terms of fidelity. This is very likely because it treats the classifier as a black-box, while the other three systems access its hidden representations. This strongly indicates that accessing hidden layers can be beneficial for fidelity of interpreters. FLINT achieves the highest fidelity, very closely followed by L2I + Θ_{MAX} and then L2I + Θ_{ATT}. This experiment serves as a sanity check for our system, that while achieving fidelity performance comparable to state-of-the-art, we hold the advantage of providing listenable interpretations in terms of pre–learnt spectral patterns.

In Table 1 (Right), we report median faithfulness $\text{FF}_{\text{median}}$ averaged over the 5 folds for our primary system L2I + Θ_{ATT}, at different thresholds τ. Smaller τ corresponds to higher $|L_{c,x}|$, which denotes the number of components being used for generating interpretations. Thus, for Random Baseline, we report $\text{FF}_{\text{median}}$ at the lowest threshold $\tau = 0.1$, to ensure removal of maximal number of components. To recall the definition of Random Baseline, please refer to Sec. 4.1. $\text{FF}_{\text{median}}$ for L2I + Θ_{ATT} is positive for all thresholds. It is also significantly higher than the Random Baseline, indicating faithfulness of interpretations.

Audio corruption experiment: an interpretability illustration. We qualitatively illustrate that the interpretations are capable of emphasizing the object of interest and are insightful for an end-user to understand the classifier's prediction. To do so, we generate interpretations after corrupting the testing data for fold–1 in two different ways (i) either with white noise at 0dB SNR (signal-to-noise ratio), (ii) or mixing it with sample of different class. It should be noted that in both these cases the system is exactly the same as before and **not** trained with corrupted samples. Some examples, covering both types of corruptions are shared on our companion website.[3]. A detailed qualitative analysis of this experiment can be found in appendix A.5 along with discussion about interpretations from other methods in appendix A.6.

4.4 Experiments on SONYC-UST

We now discuss experiments for the urban sound tagging task from the well known Detection and Classification of Acoustic Scenes and Events (DCASE) challenge 2019 & 2020 edition.

Dataset. The DCASE task used a very challenging real-world dataset called SONYC-UST [8]. It contains audio collected from multiple sensors placed in the New York City to monitor noise pollution. It consists of eight coarse-level and 20 fine-level labels. We opt for the coarse-level labeling task that involves multi-label classification into: 'engine', 'machinery-impact', 'non-machinery-impact', 'powered-saw', 'alert-signals', 'music', 'human-voice', 'dog'. This task is highly challenging for several reasons: (i) since it is real-world audio, the samples contain a very high level of background noise, (ii) the audio sources corresponding to the classes are often weak in intensity, as they are not necessarily close to the sensors, (iii) some classes may also be highly localized in time and more challenging to detect, (iv) lastly, noisy audio also makes it difficult to annotate, leading to labeling noise. This is especially true for training data that was labeled by volunteers.

Classifier performance. Our fine-tuned classifier achieves a macro-AUPRC (official metric for DCASE 2020 challenge) of 0.601. This is higher than the DCASE baseline performance of 0.510 and comparable to the best performing system macro-AUPRC of 0.649 [2]. Note that it is obtained without use of data augmentation or additional strategies to improve performance.

[3] `https://jayneelparekh.github.io/listen2interpret/`

Quantitative results. In Table 2, we report the macro-AUPRC, micro-AUPRC and max-F1 for the interpreter output w.r.t classifier. For fairness, we ignore the class 'non-machinery impact' from all class-wise evaluations involved in fidelity (*i.e.* macro-AUPRC) or faithfulness. This is because the classifier predicts only one sample in test set with positive label for this class, causing AUPRC scores to vary widely for different interpreters. VIBI has the worst performance on all three metrics for this dataset as well. In contrast to ESC-50, here the best performing system is L2I + Θ_{ATT} followed by L2I + Θ_{MAX}, and FLINT performing worse than both. The fidelity results on ESC-50 and SONYC-UST jointly demonstrate that our interpreter can generate high-fidelity *post-hoc* interpretations. Moreover, its design is flexible w.r.t different pooling functions.

The results for class-wise faithfulness are illustrated in Fig. 2a. We show $\text{FF}_{\text{median}}$ (absolute drop in probability) for our system and the Random Baseline. The results indicate that, for most classes, interpretations can be considered faithful, with a significantly positive median compared to random baseline results, which are very close to 0.

Qualitative observations. Qualitatively, we observe good interpretations for classes 'alert-signal', 'dog' and 'music'. For them, the background noise is significantly suppressed and the interpretations mainly focus on the object of interest. Interpretations for class 'human' are also able to suppress noise to a certain extent and focus on parts of human voices. However, for this class, we found presence of some signal from other audio sources too. For the remaining classes, namely 'Engine', 'Powered-saw' and 'Machinery-impact' the quality of the interpretation is more sample dependent. This is due to their acoustic similarity with the background noise. We provide example interpretations for SONYC-UST on our companion website.[3] We present an additional visualization to demonstrate coherence of our interpretations in appendix A.5.

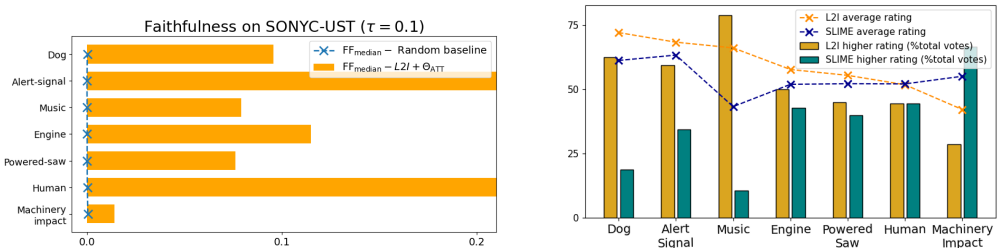
Subjective evaluation. We perform a user study (15 participants) to evaluate quality and understandability of interpretations for L2I against SLIME on SONYC-UST test data. It is worth emphasizing that understandability is one important aspect of an interpretation but is not necessarily related to its faithfulness, which should be evaluated separately, for example as proposed in Sec. 4.1. As discussed earlier, SLIME is not suitable for comparison on our quantitative metrics. Nevertheless, it is the only relevant baseline for qualitative study of listenable interpretations. Details about SLIME implementation are in appendix A.7. A participant was provided with 10 input samples, a predicted class by the classifier for each sample and the corresponding interpretation audios from SLIME and L2I. They were asked to rate the interpretations on a scale of 0-100 for the following question: "*How well does the interpretation correspond to the part of input audio associated with the given class?*". The 10 samples were randomly selected from a set of 36 (5-6 random test examples per class). For each sample, we ensured that the predicted class was both, present in the ground-truth and audible in input. Class-wise preference results and average ratings are shown in Fig. 2b. L2I is preferred for 'music', 'dog' & 'alert-signal', SLIME is preferred for 'machinery-impact', no clear preference for others. A t-test with null hypothesis that the favourable system has a lower mean score yielded p-value < 0.005 for 'music', < 0.05 for 'dog' and 'machinery-impact' and < 0.1 for 'alert-signal'.

4.5 Limitations

Finally, we list below some limitations of this study: (a) Tuning the hyperparameters requires some experience with deep architectures and audio. (b) We use phase of original input spectrogram for time-domain inversion. One could employ a phase estimation algorithm to possibly improve over this strategy. (c) The current experiments are on two datasets and one network architecture. The design of the interpreter is dependent on task and architecture of base network. Our current design of Ψ was proposed keeping in mind interpretations for a CNN operating on spectrogram-like representations. Nevertheless, it should be appropriately experimented with and modified when applying on any new data or network architecture.

5 Conclusion

To sum up, we have presented a system for post-hoc interpretation of networks that process audio. We posit that generating interpretations in terms of high-level audio objects and making them listenable are important attributes to aid understanding. Novel usage of NMF within our interpreter helps us satisfy both aforementioned requirements. Our original loss function formulation enables linking a classifier's decision to importance values over pre-learnt NMF spectral dictionary through an intermediate encoding. We perform extensive evaluation over popular audio event analysis datasets. We present a first-of-its-kind faithfulness evaluation for our non–attribution based method. Finally, a



(a) (b)

Figure 2: (a) Faithfulness results (drop in probability) for SONYC-UST arranged class-wise for threshold, $\tau = 0.1$ (b) Subjective evaluation results. Average scores for L2I and SLIME and fraction of votes in favour of each system are reported.

	Fidelity		
System	macro-AUPRC	micro-AUPRC	max-F1
L2I + Θ_{ATT}	0.909 ± 0.011	0.917 ± 0.008	0.847 ± 0.010
L2I + Θ_{MAX}	0.866 ± 0.014	0.913 ± 0.012	0.840 ± 0.012
FLINT	0.816 ± 0.013	0.907 ± 0.011	0.825 ± 0.012
VIBI	0.608 ± 0.027	0.575 ± 0.019	0.549 ± 0.020

Table 2: Fidelity results on SONYC-UST multi-label urban sound tagging task. We report AUPRC-based metrics and max F1 score for the interpreter w.r.t classifier's output (over three runs).

user study confirms usefulness of our listenable interpretations. Modular design of our system calls for further experimenting with decoder and other block architectures. We hope our work facilitates future research into designing modality-specific interpreters that aid understanding.

Acknowledgments and Disclosure of Funding

This work has been funded by the research chair Data Science & Artificial Intelligence for Digitalized Industry and Services (DSAIDIS) of Télécom Paris. This work also benefited from the support of French National Research Agency (ANR) under reference ANR-20-CE23-0028 (LIMPID project). The authors would like to thank anonymous reviewers for their valuable comments and suggestions.

References

[1] David Alvarez-Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7775–7784, 2018.

[2] Augustin Arnault and Nicolas Riche. CRNNs for urban sound tagging with spatiotemporal context. Technical report, DCASE2020 Challenge, October 2020.

[3] Roland Badeau and Tuomas Virtanen. Nonnegative matrix factorization. *Audio Source Separation and Speech Enhancement*, pages 131–160, 2018.

[4] Seojin Bang, Pengtao Xie, Heewook Lee, Wei Wu, and Eric Xing. Explaining a black-box by using a deep variational information bottleneck approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11396–11404, 2021.

[5] Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Interpreting and explaining deep neural networks for classification of audio signals. *arXiv preprint arXiv:1807.03418*, 2018.

[6] Nancy Bertin, Roland Badeau, and Gaël Richard. Blind signal decompositions for automatic transcription of polyphonic music: Nmf and k-svd on the benchmark. In *2007 IEEE Interna-*

tional Conference on Acoustics, Speech and Signal Processing-ICASSP'07, volume 1, pages I–65. IEEE, 2007.

[7] Victor Bisot, Romain Serizel, Slim Essid, and Gaël Richard. Feature learning with matrix factorization applied to acoustic scene classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1216–1229, 2017.

[8] Mark Cartwright, Ana Elisa Mendez Mendez, Jason Cramer, Vincent Lostanlen, Graham Dove, Ho-Hsiang Wu, Justin Salamon, Oded Nov, and Juan Bello. SONYC urban sound tagging (SONYC-UST): A multilabel dataset from an urban acoustic sensor network. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, pages 35–39, October 2019.

[9] Mark Cartwright, Jason Cramer, Ana Elisa Mendez Mendez, Yu Wang, Ho-Hsiang Wu, Vincent Lostanlen, Magdalena Fuentes, Graham Dove, Charlie Mydlarz, Justin Salamon, et al. Sonyc-ust-v2: An urban sound tagging dataset with spatiotemporal context. *arXiv preprint arXiv:2009.05188*, 2020.

[10] Shreyan Chowdhury, Verena Praher, and Gerhard Widmer. Tracing back music emotion predictions to sound sources and intuitive perceptual qualities. *arXiv preprint arXiv:2106.07787*, 2021.

[11] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on nmf decomposition. In *DAFx*, pages 187–194, 2014.

[12] Cédric Févotte, Emmanuel Vincent, and Alexey Ozerov. Single-channel audio source separation with nmf: divergences, constraints and algorithms. *Audio Source Separation*, pages 1–24, 2018.

[13] Ruohan Gao, Rogerio Feris, and Kristen Grauman. Learning to separate object sounds by watching unlabeled video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 35–53, 2018.

[14] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[15] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9277–9286, 2019.

[16] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[17] JL Goldstein. Auditory nonlinearity. *The Journal of the Acoustical Society of America*, 41(3): 676–699, 1967.

[18] Verena Haunschmid, Ethan Manilow, and Gerhard Widmer. audiolime: Listenable explanations using source separation. *arXiv preprint arXiv:2008.00582*, 2020.

[19] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.

[20] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pages 2127–2136. PMLR, 2018.

[21] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279*, 2017.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 326–330. IEEE, 2018.

[24] Jonathan Le Roux, John R Hershey, and Felix Weninger. Deep nmf for speech separation. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 66–70. IEEE, 2015.

[25] Jonathan Le Roux, Felix J Weninger, and John R Hershey. Sparse nmf–half-baked or well done? *Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA, Tech. Rep., no. TR2015-023*, 11:13–15, 2015.

[26] Daniel Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2001. URL `https://proceedings.neurips.cc/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf`.

[27] AM Liberman, Franklin S Cooper, Donald P Shankweiler, and Michael Studdert-Kennedy. Why are speech spectrograms hard to read? *American Annals of the Deaf*, pages 127–133, 1968.

[28] Maaten LJPvd and GE Hinton. Visualizing high-dimensional data using t-sne. *J Mach Learn Res*, 9(2579-2605):9, 2008.

[29] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[30] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016.

[31] Saumitra Mishra, Bob L Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *ISMIR*, pages 537–543, 2017.

[32] Saumitra Mishra, Emmanouil Benetos, Bob LT Sturm, and Simon Dixon. Reliable local explanations for machine listening. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[33] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[34] Hannah Muckenhirn, Vinayak Abrol, Mathew Magimai-Doss, and Sébastien Marcel. Understanding and visualizing raw waveform-based cnns. In *Interspeech*, pages 2345–2349, 2019.

[35] Jayneel Parekh, Pavlo Mozharovskyi, and Florence d'Alché Buc. A framework to learn with interpretation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[36] Sanjeel Parekh, Alexey Ozerov, Slim Essid, Ngoc QK Duong, Patrick Pérez, and Gaël Richard. Identify, locate and separate: Audio-visual object extraction in large video collections using weak supervision. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 268–272. IEEE, 2019.

[37] Geoffroy Peeters and Gaël Richard. Deep learning for audio and music. In J. Benois-Pineau and A. Zemmari, editors, *Multi-faceted Deep Learning: Models and Data*, pages 231–266. Springer, 2021.

[38] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[39] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[40] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[41] Michael Schoeffler, Sarah Bartoschek, Fabian-Robert Stöter, Marlene Roess, Susanne Westphal, Bernd Edler, and Jürgen Herre. webmushra—a comprehensive framework for web-based listening tests. *Journal of Open Research Software*, 6(1), 2018.

[42] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2019.

[43] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[44] Paris Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *International Conference on Independent Component Analysis and Signal Separation*, pages 494–499. Springer, 2004.

[45] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[46] Vincent YF Tan and Cédric Févotte. Automatic relevance determination in nonnegative matrix factorization with the/spl beta/-divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1592–1605, 2012.

[47] Sana Tonekaboni, Shalmali Joshi, Kieran Campbell, David K Duvenaud, and Anna Goldenberg. What went wrong and when? instance-wise feature importance for time-series black-box models. *Advances in Neural Information Processing Systems*, 33:799–809, 2020.

[48] Kevin W Wilson, Bhiksha Raj, Paris Smaragdis, and Ajay Divakaran. Speech denoising using nonnegative matrix factorization with priors. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4029–4032. IEEE, 2008.

[49] Scott Wisdom, Thomas Powers, James Pitton, and Les Atlas. Deep recurrent nmf for speech separation by unfolding iterative thresholding. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 254–258. IEEE, 2017.

[50] Minz Won, Sanghyuk Chun, and Xavier Serra. Toward interpretable music tagging with self-attention. *arXiv preprint arXiv:1906.04972*, 2019.

[51] Chih-Kuan Yeh, Been Kim, Sercan O Arik, Chun-Liang Li, Pradeep Ravikumar, and Tomas Pfister. On concept-based explanations in deep neural networks. *arXiv preprint arXiv:1910.07969*, 2019.

[52] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Invase: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

[53] Pablo Zinemanas, Martín Rocamora, Marius Miron, Frederic Font, and Xavier Serra. An interpretable deep learning model for automatic sound classification. *Electronics*, 10(7):850, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We discuss potential societal impacts in appendix A.9

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] All assets are free and public for research
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] Please see appendix A.8
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] It was optional for participants

A Appendix

A.1 Sparse-NMF implementation details

The pre-specified dictionary (Step 2 in Algorithm 1) is learnt using Sparse-NMF [25]. To recall, the following optimization problem is solved through multiplicative updates to pre-learn $\mathbf{W}$:

$$\min D(\mathbf{X}_{\text{train}}|\mathbf{W}\mathbf{H}) + \mu\|\mathbf{H}\|_1 \quad s.t. \mathbf{W} \geq 0, \mathbf{H} \geq 0, \|\mathbf{w}_k\| = 1, \ \forall k. \tag{7}$$

Training audio files are converted into log-magnitude spectrogram space for factorization. We construct $\mathbf{X}_{\text{train}}$ differently for each dataset due to their specific properties. For ESC-50, $\mathbf{X}_{\text{train}}$ is constructed by concatenating the log–magnitude spectrograms corresponding to each sample in the training data of the cross-validation fold (1600 samples for each fold) and performing joint factorization using Eq. 7.

SONYC-UST however, is an imbalanced multilabel dataset with very strong presence of background noise. A typical procedure to learn components, as for ESC-50, yields many components capturing significant background noise. This affects understandability of interpretations. As a result, we process this dataset differently. We first learn $\mathbf{W}_{\text{noise}}$, that is, a set of 10 components to model noise using training samples with no positive label. Then, for each class, we randomly select 700 positively-labeled samples from all training data and learn 10 new components (per class) with $\mathbf{W}_{\text{noise}}$ held fixed for noise modeling. All $10 \times 8 = 80$ components are stacked column-wise to build our dictionary $\mathbf{W}$. While this strategy helps us reduce the number of noise-like components in the final dictionary, it does not completely avoid it.

As done in [7], for computational efficiency, we too average the spectrogram frames over chunks of five. This reduces the size of $\mathbf{X}_{\text{train}}$ and saves memory to allow training over more number of samples.

A.2 Classifier f details

The architecture we use for f [23] has been pretrained on AudioSet. For each dataset, we first fine-tune this network and perform post-hoc interpretations for the resulting trained network. Here we discuss its broad architecture and specific training details used to fine-tune it on our datasets.

It takes as input a log-mel spectrogram. The architecture broadly consists of six convolutional blocks (B1–B6) and one convolutional layer with pooling for final prediction. Most convolutional blocks consist of two sets of conv2D + batch norm + ReLU layers followed by a max pooling layer.

Details of the full architecture can be found in the original reference. For fine-tuning, we modify the architecture of prediction layers. Specifically, we remove the F2 conv layer and add a linear layer after final pooling, the output dimensions of which correspond to the number of classes in our datasets.

For both the datasets, we do not use any data augmentation. The ADAM optimizer [22] is used to fine-tune f. For ESC-50, we only fine-tune the prediction layers of the network. We train the classifier for 10 epochs on each fold of the dataset with a learning rate of 1×10^{-3}.

On SONYC-UST, we fine-tune all the layers in f, which leads to higher classifier AUPRC metrics. The classifier is trained for 10 epochs. Here we start with a learning rate of 2×10^{-4} and halve it after every 4 epochs.

A.3 Choosing number of components K

Choice of number of components, K, also known as order estimation, is typically data and application dependent. It controls the granularity of the discovered audio spectral patterns. Choosing K has also been a long standing problem within the NMF community [46]. Our choice for this parameter was guided by three main factors:

- Choices made previously in literature for similar pre-learning of $\mathbf{W}$ [7], who demonstrated reasonable acoustic scene classification results with a dictionary size of $K = 128$. We used this as a reference to guide our choice for number of components.
- Dataset specific details which include number of classes, samples for each class, variability of recordings etc. For eg. acoustic variability of ESC-50 (larger number of classes), prompted us to use a dictionary of larger size compared to SONYC-UST.
- When tracking loss values for different K, we observed a plateauing effect for larger dictionary sizes as illustrated in Fig. 3 for ESC-50.

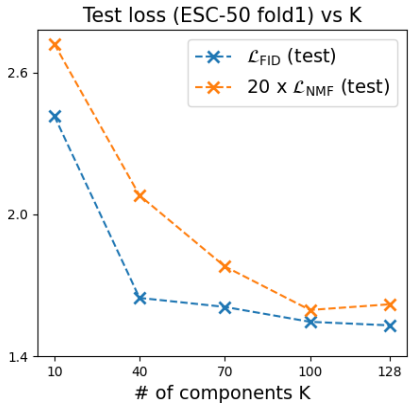


Figure 3: Loss values on ESC50-test data for fold 1 for various dictionary sizes.

ConvBlocks	$\mathcal{L}_{\text{NMF}}$	$\mathcal{L}_{of}$	top-1
B4+B5+B6	**0.079**	**1.546**	**65.5**
B5+B6	0.103	1.572	61.5
B6	0.118	1.698	57.8
Input	0.102	2.384	34.5

Table 3: Hidden layer ablation study (ESC-50). Current choice indicated in bold.

α	β	$\mathcal{L}_{\text{NMF}}$	$\mathcal{L}_{of}$	macro-AUPRC
10.0	**0.8**	**0.028**	**0.386**	**0.900**
10.0	8.0	0.048	0.386	0.879
10.0	0.08	0.028	0.388	0.876
1.0	0.8	0.045	0.375	0.921
100.0	0.8	0.027	0.445	0.612

Table 4: Loss hyperparameter ablation study on SONYC-UST. Current choice in bold.

A.4 Other hyperparameters and ablation studies

Audio processing parameters. For both the tasks, we perform same audio pre-processing steps. All audio files are sampled at 44.1kHz. STFT is computed with a 1024-pt FFT and 512 sample hop size, which corresponds to about 23ms window size and 11.5ms hop. The log-mel spectrogram is extracted using 128 mel-bands.

Other hyperparameters We used the same set of hidden layers for both datasets. Specifically, we use the outputs of last three convolutional blocks in f, B4, B5 and B6. We also used the same loss hyperparameters $\alpha = 10, \beta = 0.8$ for both datasets. Models were optimized using ADAM [22] for 35 epochs on each fold of ESC-50 with learning rate: 2×10^{-4} and for 21 epochs on SONYC-UST (learing rate: 5×10^{-4}).

Tab. 3 and Tab. 4 present ablation studies for loss hyperparameters and choice of hidden layers. The choices in bold indicate our current choices. The metrics and loss values given here are for a single run. For the ablation study on hidden layers in Tab. 3, we additionally report another baseline where instead of accessing the hidden layers, Ψ is directly applied on the input. Given that the interpreter no longer has access to representations learnt by the classifier (which were close to the output as well) and architecture of Ψ itself is much simpler compared to the classifier, it is significantly worse at approximating classifiers output.

Total training time is around 50 minutes for 1 fold on ESC-50 and 150 minutes for SONYC-UST. Around 30-40% of the total time is spent on pre-learning $\mathbf{W}$ using Sparse-NMF (for both datasets). Networks were trained on a single NVIDIA-K80 GPU.

A.5 Further discussion on Interpretations

A.5.1 Corruption samples ESC-50

The goal of this experiment is to qualitatively illustrate that our method can generate interpretations on ESC-50 in various noisy situations. For this, we corrupt a given sample from a target class in two ways: (i) With sample from a different class (Overlap experiment), and (ii) Adding high amount of white noise, at 0dB SNR (Noise experiment). The key question that we want the interpretations to offer insight on is: *did the classifier truly make its decision because it "heard" the target class or is it making the decision based on the corruption part of the audio?* The cases where classifier misclassifies are analyzed in Sec. A.5.2. As already highlighted in Sec. 1, listenable interpretations are not expected to perform source separation for the class of interest, but to confirm if decision corresponds entirely/mostly to target class or not. All examples can be listened to on our companion website [4]. Since the target and corrupting signals and their classes are already known, we can reinforce the observations drawn by listening to the interpretations through spectrograms (Figs. 4, 5).

A.5.2 Misclassification samples ESC-50

When the classifier prediction is incorrect, the interpretations may still provide insight into the classifier's decision by indicating what the classifier "heard" in the input signal. We give examples for this on the webpage[4]. For instance, one of the example is of a sample with ground-truth class 'Crying-Baby' misclassified as a 'Car-horn'. Interestingly, the interpretation is acoustically similar to car horns. Please note the importance of *listenable* interpretations that aid such understanding into the audio network's decisions.

[4] `https://jayneelparekh.github.io/listen2interpret/`

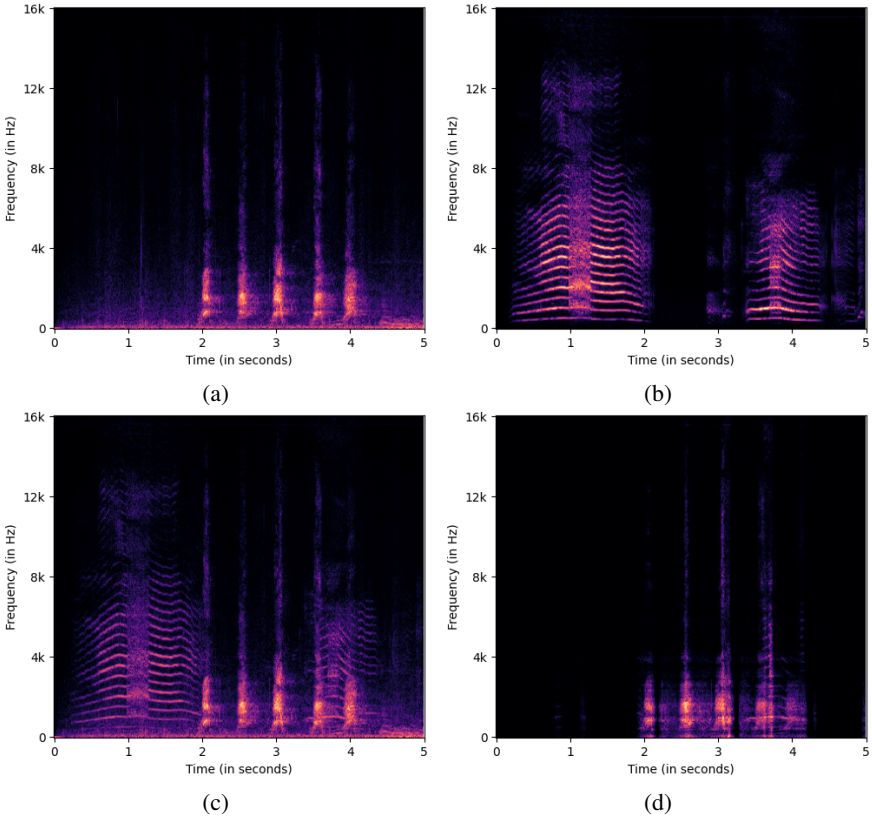


Figure 4: Log-magnitude spectrograms of an example from Overlap experiment: (a) Target class ('Dog') original uncorrupted signal (b) Corrupting/Mixing class ('Crying-Baby') signal (c) Corrupted/mixed signal, also the input audio to the classifier (d) Interpretation audio for the predicted class ('Dog'). The interesting observation is that spectrogram of interpretation audio almost entirely consists of parts from target class ('Dog') signal with only a very weak presence of corrupting class ('Crying-Baby') close to the end.

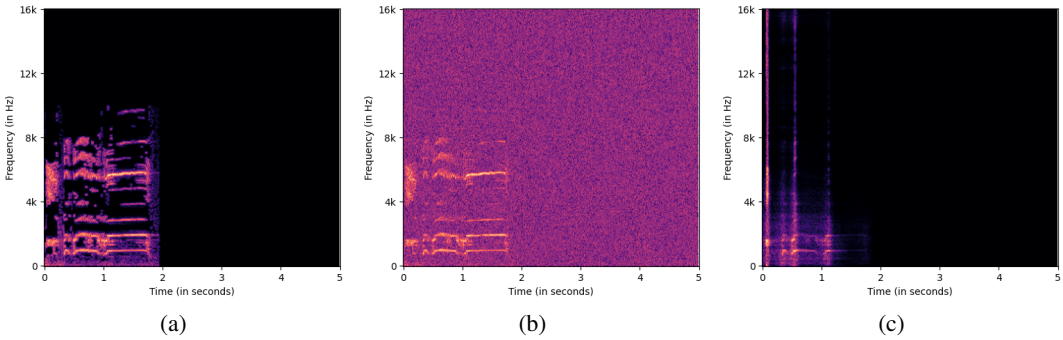


Figure 5: Log-magnitude spectrograms of an example from ESC-50 Noise experiment: (a) Target class ('Rooster') original uncorrupted signal, (b) White noise corrupted signal, also the input audio to the classifier (c) Interpretation audio for the predicted class ('Rooster'). Again, the interpretation audio is almost entirely free of corrupting signal (white noise in this case) and mostly consists of parts of the original target signal. This strongly indicates that the classifier relied on parts of audio corresponding to the target class to make its decision, and not the white noise.

A.5.3 Coherence in interpretations

We qualitatively analyze the interpretations on SONYC-UST by visualizing relevances generated on the test set. Specifically, we compute the vector $r_{c,x} \in \mathbb{R}^K$ which contains relevances of all components in prediction for class c for sample x. The relevance vectors are collected for each test sample x and its predicted class c. We then apply a t-SNE [28] transformation to 2D for visualization. This is shown in Fig. 6. Each point is colored according to the class for which we generate the interpretation. Interpretations for any single class are coherent and similar to each other. This is to some extent a positive consequence of global weight matrix in Θ. Moreover, globally it can be observed that classes like 'Machinery-impact' and 'Powered-Saw' have similar relevances which are to some extent close to 'Engine'. This is to be expected as these classes are acoustically similar. 'Dog' and 'Music' are also close in this space, likely due to the often periodic nature of barks or beats.

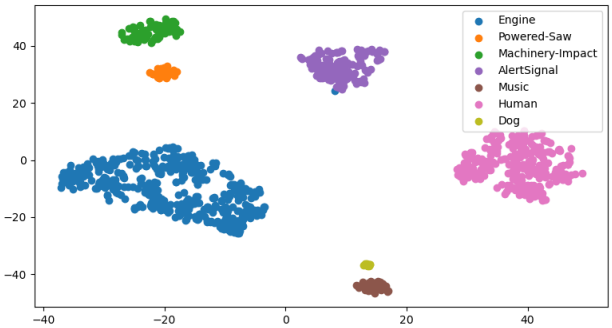


Figure 6: Visualized relevances (following a t-SNE transformation) of generated interpretations on SONYC-UST, colour-coded according to interpreted class.

A.6 Discussion on interpretations from related methods

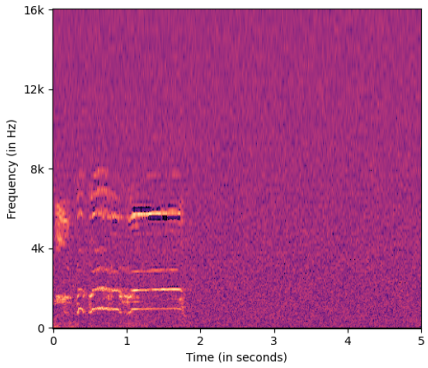
A.6.1 Attribution maps for listenable output

Input attribution/saliency maps in their current form are more suitable for images. These maps are generally spatially smooth, which aids visual understandability, but are not effective masks to clearly emphasize time-frequency bins. Thus, for audio spectrogram like inputs, while they can be useful in visually indicating the important regions, they are poor masks to filter such information for listenable output. We applied a recent approach based on information bottleneck [42] to generate attribution maps for few samples on ESC50-Noise Experiment.

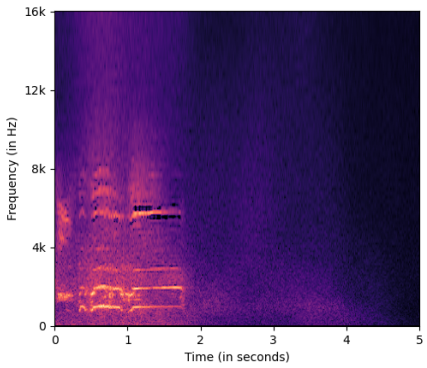
Experimental details: We used the python PyTorch version of their package and follow the standard example version given in their repository [5]. The example inserts a bottleneck in conv layer from 4th block of VGG16. Our network architecture is also similar to VGG architectures. So we applied a bottleneck at the output of 4th conv block (B4), which we also access via our interpreter. We also follow the same optimization procedure as in the example, i.e. Adam for 10 iterations. The saliency map is applied as a filter on the mel-spectrogram. We then approximate STFT from mel-spectrogram and invert it using input phase for a time-domain audio output.

Outputs can be heard on our companion website [4]. We provide visualizations for a sample in Fig. 7. While the saliency map indeed visually indicates relevant regions, the time-domain signal still contains considerable noise and is not very useful. The smoothness of saliency maps can be partly attributed to upsampling of information extracted from lower resolution feature maps. Another limitation of applying these methods to 2D CNN's is the frequent use of log-mel spectrogram as input (current model uses 128 mel bands) for the networks. The saliency map is then over the mel-spectrogram space. This adds to the loss of information and exacerbates issues in their use as filtering masks for spectrograms. Despite their usefulness, we believe these methods require non-trivial updates to be suitable for generating listenable interpretations.

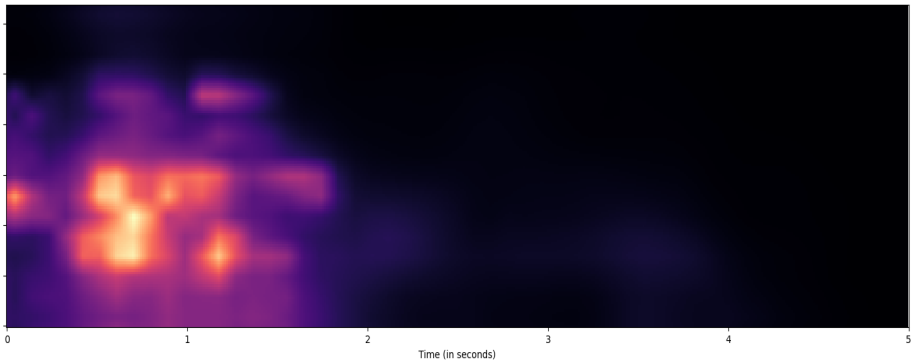
[5] `https://github.com/BioroboticsLab/IBA`



(a)



(b)



(c)

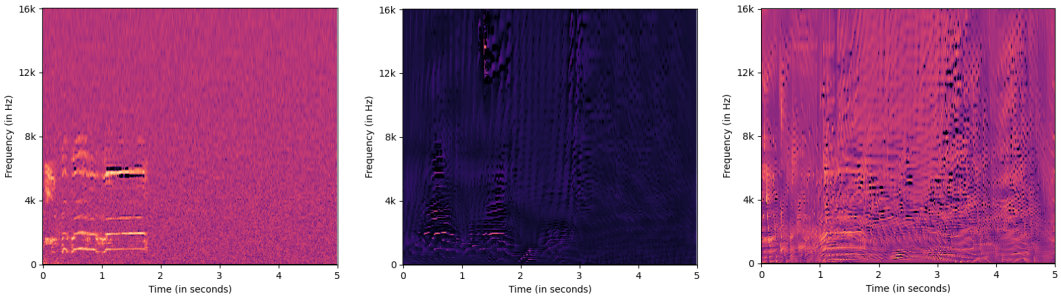
Figure 7: Log-magnitude spectrograms and saliency map to visualize an attribution map on ESC50-Noise sample: (a) White noise corrupted signal (from class 'Rooster'), also the input audio to the classifier, (b) Interpretation audio for the predicted class ('Rooster'), (c) Saliency map on the log-mel spectra space. The regions corresponding to the signal frequencies are brightest in the saliency map. However, owing to it's smoothness and loss of information in mel-spectrogram space, high amount of noise is still a part of interpretation signal

A.6.2 Interpretations of FLINT

For completeness, we also provide examples of interpretations by FLINT on ESC-50 Noise samples. As discussed in Sec. 2, FLINT uses a visualization pipeline to understand high-level attributes, which primarily consists of using activation maximization [30] based procedure to emphasize patterns relevant for the activation of an attribute.

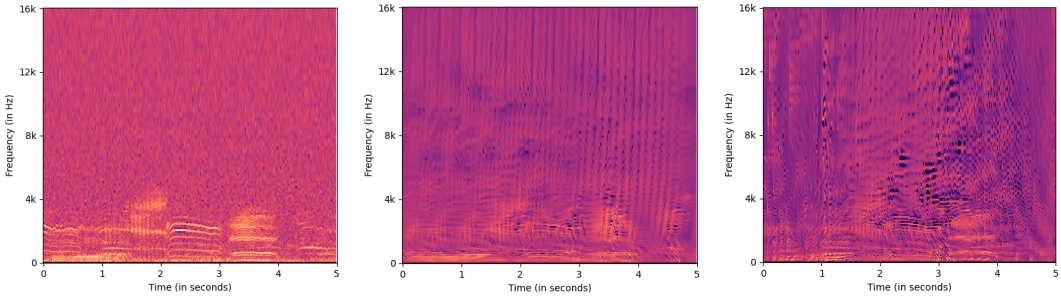
In our current setting, this optimization procedure takes place in the log-mel spectrogram space. For initialization with a "weak version" version of the input we subtract 10 from the input log-mel spectrogram. We use Adam optimizer for 1500 iterations We add below examples of this visualization strategy after estimating log-magnitude spectrogram from the output of optimization procedure. Additionally we also estimate the time-domain signal as before to verify any potential as listenable output on the webpage [4].

The optimization in general results in specific patterns added in a log mel-spectrogram and thus the magnitude spectrogram. However, visually understanding the significance of the patterns is a very hard task. Listening to the resulting spectrograms is not informative either as they typically do not remove the noise, nor do they correspond to recognizable phenomenon. Compared to dictionary of pre-learnt spectral patterns, the dictionary of attributes is less constrained in the information an individual attribute encodes. Moreover, FLINT's visualization pipeline provides finer-grained interpretation at an attribute level. Both these considerations require the pipeine to be lot more effective to convey the interpretation understandably for audio modality.



(a) (b) (c)

Figure 8: Log-magnitude spectrogram visualizations for two relevant attributes of FLINT on a sample from ESC50-Noise experiment: (a) White noise corrupted input audio (class: 'Rooster'), (b) Activation maximization output for attribute 62, (c) Activation maximization output for attribute 77.



(a) (b) (c)

Figure 9: Log-magnitude spectrogram visualizations for two relevant attributes of FLINT on a sample from ESC50-Noise experiment: (a) White noise corrupted input audio (class: 'Sheep'), (b) Activation maximization output for attribute 7, (c) Activation maximization output for attribute 77.

A.7 Baseline implementations details

FLINT: We implemented it with the help of their official implementation available on GitHub.[6] For each experiment, we fix their number of attributes J equal to the number of our NMF components K. We also choose the same hidden layers for their system as we choose for ours. This baseline is trained for the same number of epochs as us. We use same values for our $\mathcal{L}_{\text{NMF}}$ loss weight, α, and their $\mathcal{L}_{if}$ loss weight γ. For the other loss hyperparameters, we use their default values and training strategy.

VIBI: We implemented this using their official repository.[7] The key hyperparameters that we set are the input chunk size and their parameter K, the number of chunks to use for interpretation. We use a larger chunk size than in their experiments to limit the number of chunks. On ESC-50, we use a chunk size of 32×43, and on SONYC-UST, a chunk size of 32×86. This yields 40 chunks for each input on both the datasets. We varied the K from 5 to 20, and report the results with best fidelity. The system was trained for 100 epochs on ESC-50 and 30 epochs on SONYC-UST

SLIME: We primarily relied on implementation from their robustness analysis repository [8]. The key hyperparameters to balance are the number of chunks vs chunk size. SONYC-UST contains 10 second audio files. This is much longer than 1.6 second audio files for which SLIME was originally demonstrated [31]. Therefore, we divide only on the time-axis to limit the number of chunks. SLIME recommends a chunk size of at least 100ms. They operate on upto 290ms chunk size. We balance these two hyperparameters by dividing our audio files in 20 chunks of 500ms chunk size. We select a maximum of 5 chunks for interpretations and a neighbourhood size of 1000.

[6] `https://github.com/jayneelparekh/FLINT`
[7] `https://github.com/SeojinBang/VIBI`
[8] `https://github.com/saum25/local_exp_robustness`

A.8 Subjective evaluation implementation

The subjective evaluation interface was implemented using webMUSHRA [41]. Prior to voting on the test samples, participants were provided with an instruction page and then a training page with an example to get used to interface, instructions, tune their volume etc. Screenshots of the instruction and training page are given in Fig. 10, Fig. 11 respectively.

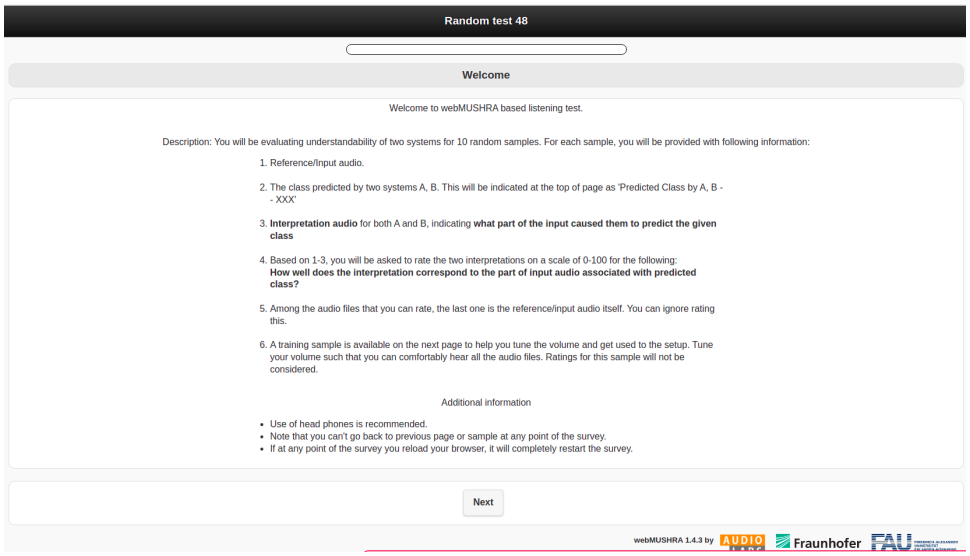


Figure 10: Instructions for the participants at the start of the subjective evaluation

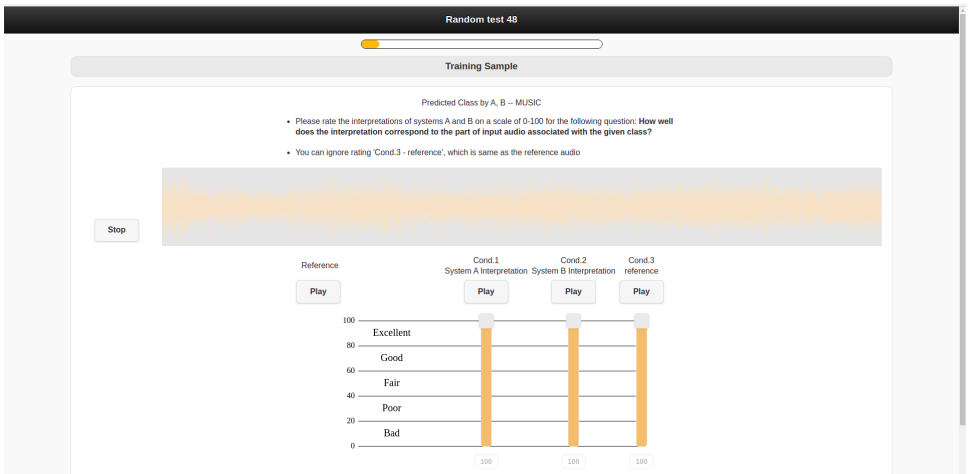


Figure 11: Training page for subjective evaluation that illustrates the interface for scoring for the participants.

A.9 Potential Societal Impacts

We expect our method to have positive societal impact by improving understandability of interpretations for audio processing networks. However, this inherently benign technology could be misused when in wrong hands. For example, it can be used to provide misleading interpretations if trained incorrectly (wrong NN architectures, insufficient training examples/training epochs, malicious datasets etc.). Evidently, we expect proper use of the developed methodology, although direct misuse protection mechanisms were not developed in this piece of research, not being the initial goal.