Machine Learning Group


— Master's Thesis —

# Disentangled Explanations for Neural Network Predictions on Audio Data

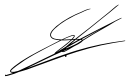
**Author**: Samuel Harck

**Supervisor**: Prof. Klaus-Robert Müller
Prof. Grégoire Montavon

**Place and Date**: Berlin, October 18, 2024

# Declaration of Originality

I hereby certify that I have prepared this thesis without the help of third parties and without the use of sources and aids other than those indicated. The passages taken verbatim or in terms of content from the sources used are identified as such. I confirm that this assignment is my own work, is not copied from any other person's work (published or unpublished), and has not previously been submitted for assessment.

| 2. Juli 2024 |  |
| --- | --- |
| Date & Time | Signature |

# Abstract

As nonlinear *Machine Learning* (ML) models are increasingly used in various real world applications, their black-box nature hinders in-depth model evaluation, apart from performance measures. In response, the field of *Explainable Artificial Intelligence* (XAI) has made much progress. It aims to reveal the rationale behind complex ML models, often by assigning relevance scores to model parts and input features, e.g., pixels. However, in some domains such as audio processing, where data—like time or time-frequency representations of amplitudes—is of rather unintuitive nature, the extracted explanations can be hard to grasp for humans. Suitably, a novel sub-field in the realm of XAI has emerged in very recent time that aims to decompose explanations into multiple sub-explanations, representing distinct decision concepts. These approaches offer a promising foundation to gain more valuable insights into models and the data domain, especially when dealing with complex data scenarios. This study targets the extraction of concept-based explanations for a neural network applied to audio classification tasks, by utilizing the newly proposed method, *Disentangled Relevant Subspace Analysis* (DRSA), in combination with *Layer-wise Relevance Propagation* (LRP).

# Contents


**1 Introduction** **1**

1.1 Motivation . . . 1

1.2 Outline . . . 3

**2 Theoretical Background** **4**

2.1 Neural Networks . . . 4

2.1.1 Backpropagation . . . 5

2.1.2 Convolutional Neural Networks . . . 6

2.2 Explainable Artificial Intelligence . . . 8

2.2.1 Attribution-based Explanation Techniques . . . 9

2.2.2 Concept-based Explanation Techniques . . . 12

2.2.3 Evaluating Explanations . . . 15

2.3 Audio Processing . . . 16

2.3.1 Audio Representations . . . 16

2.3.2 Music Information Retrieval . . . 21

2.3.3 Explaining Music Classifiers . . . 21

**3 Methodological Setup** **23**

3.1 Model . . . 23

3.2 Explanation Setup . . . 24

3.2.1 Local Attribution . . . 24

3.2.2 Disentangled Relevant Subspace Analysis . . . 25

3.2.3 Two-Step Attribution . . . 27

3.3 Transforming Explanations into Audios . . . 27

**4 Experiments** **30**

4.1 Synthetic Data . . . 30

4.1.1 Data Construction . . . 30

4.1.2 Setup . . . 32

4.1.3 Evaluation . . . 32

4.2 Music Showcase . . . 35

4.2.1 Setup . . . 35

4.2.2 Qualitative Evaluation . . . 36

4.2.3 Quantitative Evaluation . . . 39

**5 Conclusion** **45**

5.1 Main Findings . . . 45

5.2 Future Work . . . . . 45

**A Model Details and Training Info** **I**

A.1 Data Preprocessing . . . . . I

A.2 Model Optimization . . . . . III

A.3 Model Evaluation . . . . . IV

**B Implementation Details** **VI**

B.1 Layer-wise Relevance Propagation . . . . . VI

B.1.1 Evaluating Local Explanations . . . . . VI

B.2 Two-step Attribution and Disentangled Explanations . . . . . VII

B.2.1 Optimization of Subspaces . . . . . VII

B.2.2 Two-step Attribution . . . . . VIII

B.3 Audios from Explanations . . . . . VIII

**C Qualitative Evaluation Supplement** **IX**

C.1 Synthetic Data . . . . . IX

C.2 GTZAN Data . . . . . XI

**D Synthetic Data Generation** **XIV**

D.1 Synthetic Toy Class 1 . . . . . XIV

D.2 Synthetic Toy Class 2 . . . . . XVI

D.2.1 Final Sample Construction . . . . . XVII

# List of Figures


2.1 Convolution between two feature maps . . . . . 8
2.2 Schematic redistribution process of LRP . . . . . 10
2.3 Virtual layers trained with DRSA . . . . . 14
2.4 Audio representations . . . . . 19
2.5 Mel filter bank . . . . . 20

3.1 Architectural framework of the model . . . . . 23

4.1 Synthetic toy class 1 . . . . . 31
4.2 Synthetic toy class 2 . . . . . 31
4.3 Disentangled explanations for synthetic toy class 1, Conv3 . . . . . 33
4.4 Class specific audio objects of toy class 1 . . . . . 33
4.5 Disentangled explanations on synthetic class 1 with random subspaces . . . 34
4.6 DRSA results on genre class hiphop . . . . . 38
4.7 DRSA results on genre class jazz . . . . . 38
4.8 Qualitative experiment on class distinctiveness of concept subspaces . . . . 39
4.9 Visualization of the patch flipping procedure . . . . . 40
4.10 Cross-class patch-flipping scores . . . . . 43

A.1 Training curves of model optimization . . . . . V
A.2 Confusion matrix of GTZAN model . . . . . V

B.1 Patch-flipping evaluation of LRP . . . . . VII
B.2 Learning curves DRSA . . . . . VII
B.3 Histogram plot of relevance distribution . . . . . VIII

C.1 Disentangled explanations on toy class 2, Conv3 . . . . . IX
C.2 Class specific audio objects of synthetic class 2 . . . . . IX
C.3 Disentangled explanations on toy class 2 . . . . . X
C.4 DRSA results on genre class reggae . . . . . XI
C.5 DRSA results on genre class metal . . . . . XII
C.6 DRSA results on genre class classical . . . . . XII
C.7 DRSA results on genre class hiphop, 2 subspaces . . . . . XIII
C.8 DRSA results on genre class hiphop, 8 subspaces . . . . . XIII

D.1 Class specific audio objects of toy class 1 . . . . . XV
D.2 Class specific audio objects of toy class 2 . . . . . XVI

# List of Tables

4.1 Sound objects contained in the synthetic dataset . . . . . 30
4.2 Classes and samples contained in the GTZAN dataset . . . . . 35
4.3 LRP composite of the genre recognition model . . . . . 36
4.4 Patch-flipping scores for disentanglement performance . . . . . 41

A.1 Data augmentation during model training . . . . . II

# List of Abbreviations

| | |
|---|---|
| **ASR** | *Automatic Speech Recognition* |
| **ASP** | *Audio Signal Processing* |
| **AUPC** | *Area under the Pixel-Flipping Curve* |
| **CNN** | *Convolutional Neural Network* |
| **DFT** | *Discrete Fourier Transform* |
| **DL** | *Deep Learning* |
| **DNN** | *Deep Neural Network* |
| **DRSA** | *Disentangled Relevant Subspace Analysis* |
| **DSA** | *Disentangled Subspace Analysis* |
| **DTD** | *Deep Taylor Decomposition* |
| **FT** | *Fourier Transform* |
| **LRP** | *Layer-wise Relevance Propagation* |
| **MGR** | *Music Genre Recognition* |
| **MIR** | *Music Information Retrieval* |
| **ML** | *Machine Learning* |
| **NN** | *Neural Network* |
| **ReLU** | *Rectified Linear Unit* |
| **STFT** | *Short-time Fourier Transform* |
| **XAI** | *Explainable Artificial Intelligence* |

# 1 Introduction

## 1.1. Motivation

Machine learning accounts for the aim of automatizing various tasks by building predictive models that map inputs to outputs [1], [2]. While achieving outstanding performance [2], [3], [4], [5], ML systems have become integral in technologies such as image recognition [6], [4], or *Audio Signal Processing* (ASP) [7], [8]. Especially in domains that encompass intricate data environments like audio, *Deep Learning* (DL) methods such as Deep Neural Networks (DNNs) have paved their way to be the preferred choice. Specific applications in the realm of ASP include automatic speech recognition [9], [7], [10], [11], *Music Information Retrieval* (MIR) [12], [13], [14], [15], music generation [16], [17], or audio event classification [18], [19]. Since audio signals can be represented in various forms, e.g., waveforms (large sequences of amplitudes), or time-frequency representations (spectrograms that depict frequency content over time), DL models can be trained on diverse input types, delivering exceptional results [20], [14], [19], [8].

However, to account for the growing need of providing robust and trustworthy systems, where high performance is not the only characteristic of importance [21], [22], the decision strategy of the applied models requires to be unfolded. Yet, most ML approaches, especially DNNs, usually exhibit a high nonlinearity which comes along with their capability of solving complex tasks [1], [2]. This nonlinearity engenders these models to be inherently hard to interpret which is why they are often referred to as 'black-box' models. For these reasons, the field of XAI has emerged as an active research area with the goal to reveal the reasoning structure of nonlinear ML methods [23], [24], [25], [26].

In this fairly new field of research, already a variety of methods have been proposed [23], [25], [24], [27], [28], [29]. Within the realm of post-hoc explanations, i.e., explaining fully trained models, particularly attribution-based approaches have gained high popularity [30], [23], [25], [26]. Through attributing relevance scores to model parts and input features, these methods provide detailed insights into the rationale of ML models with respect to single data points, thus providing local explanations. These methods have proven to be beneficial in various tasks, which include: revealing a biased decision behaviour of ML models towards spurious correlations in data, unmasking the so called 'Clever-Hans' effect [31], [32], [33], improvement of model performance [34], [35], [36], [37], or the discovery of new scientific insights [38], [39].

Although attribution-based XAI has shown to be a powerful tool, there are still limitations associated to them [22], [40]. Through highlighting important regions, i.e., visualizing where to find relevant features, the questions remains if and how those features are interconnected to eventually form high-level objects governing a models decision [35], [41], [42], [43], [44]. Hence, it is of big interest to extract such richer explanations as these could provide semantically more valuable insights into a model and the data domain [22]. Consequently, a novel sub-field of XAI has recently gained attention that focuses on extracting richer structured explanations, especially, concept-based explanations [35], [41], [43], [42]. These methods are based on discoveries that neural networks encode latent concepts which play a significant role in their decision-making process [45], [46]. Through extracting such global concepts from hidden feature layers of DL models, yet, expressing them locally, these approaches aim to combine benefits of both, local and global XAI [35], [41].

Applying XAI to ASP systems can service different aspects. Gaining insights into models and the audio domain that is being analyzed, allows to compare ASP systems trained on different types of input representations on a higher level [47], [48], [49]. Furthermore, highlighting important features in the inputs that guide the decision making process of audio classification models can lead to a deeper understanding of the characteristics present in the data, and allow the design of more directed approaches for some task at hand [50], [51], [49], [48]. However, the utility of traditional local explanations may suffer from the inherent nature of audio data [52]. In particular, audio data such as music is typically a composition of multiple overlaid sound objects, forming the resulting signal [53], [54]. Hence, obtaining a multitude of relevance scores associated with various input features may by of limited use, by merging important components into a single explanation [55], [52]. On the account of this, concept-based XAI represents a promising approach to analyze audio data, and more generally, high dimensional data such as time-series. Speaking about *Music Genre Recognition* (MGR), decomposing explanations into several sub-explanations can lead to an increased understanding on the compositional nature of music that groups musical pieces into several genres [55], [50], [52].

Yet, ensuring that explanations are understandable to humans remains a challenge in all the aforementioned XAI techniques [51], [49]. Whereas some domains already offer a direct interpretable basis, e.g. images or text that can be overlayed with relevance scores, others lack on this advantage. This is most likely the reason why the majority of progress in XAI was made in fields with intuitively interpretable data. For instance, in the realm of time-series analysis, XAI methods are still being adapted. Interpretation of data in time domain forms a challenging task due to its high dimensionality [56], [49]. Given the prevalence of time dependent data in real world scenarios such as the medical field, forecasting applica-

tions, or the audio domain, this limitation has to be addressed. Nevertheless, the realm of audio data provides a solid foundation that is naturally comprehensible to humans, namely listenable audio tracks. This feature can be leveraged when explaining audio classification systems.

Taking one further step towards explaining *Neural Network* (NN) decisions on a higher level, this work conquers the extraction of concept-based explanations on DL methods applied in the domain of audio data. This is tackled by showcasing the performance of DRSA [35] to extract multiple sub-explanations on an audio classification task, specifically, music genre recognition. Furthermore, a methodology is established to transform explanatory components into a human interpretable domain, i.e., listenable audio tracks, to improve evaluation capabilities.

## 1.2. Outline

The core components of this work are structured into four distinct work packages. Firstly, a task related to the field of ASP, namely MGR, is solved by training a DL approach on audio inputs in time-frequency domain. Secondly, an attribution-based XAI method, e.g., LRP [23], is employed to explain model decisions locally, i.e. for single data points. Thereafter, relevant subspaces are optimized with DRSA [35] that represent important components within the data the model has encoded. After implementing a two-step attribution to visualize joint input-concept explanations at the input domain, a methodology is provided to transform explanations into listenable audio tracks. This will play in hand with the last step, namely qualitative and quantitative evaluation of the explanation performance. To provide a solid basis for evaluation, a synthetic dataset is created that fits the purpose of this study.

# 2 | Theoretical Background

## 2.1. Neural Networks

As the use of ML has seen a significant surge in recent years, tasks to solve increase in complexity, and the volume of data to process is growing [2], [57]. For these reasons, especially DL approaches gained high popularity as these methods have achieved superior performance in various tasks, including pattern recognition [4], *Automatic Speech Recognition* (ASR) [58], or natural language processing [5], [9]. The main driver for this development is their ability to independently learn powerful representations from raw data, which serve as foundation for their decision-making [59], [60].

Neural networks are predictive models that produce real valued outputs for given inputs. The core component of a NN is the artificial neuron, which is inspired by the biological neuron contained in the human brain. It can be seen as an extreme simplification of a biological neuron that only kept two major functionalities, namely to process information in a nonlinear manner, and the ability to learn. Through interconnecting a large number of artificial neurons, NNs are capable of approximating any nonlinear function, albeit depending on their size (universal approximation theorem) [1], [60], [61]. But how do neurons facilitate the ability to learn?

A neuron defines a nonlinear function that can process an arbitrary number of inputs, so called input activations, to produce some real valued output. Suppose the input activations as $a_j$, with $j$ being an index for each input. The mapping performed by a neuron can decomposed into two separate steps, beginning by calculating a weighted sum over the inputs according to

$$z_k = \sum_j a_j w_{jk} + b_k \,, \tag{2.1}$$

where $w_{jk}$ and $b_k$ define the learnable parameters of neuron $k$, and are referred to as weights, and bias, respectively. Thereafter, the weighted sum $z_k$ is typically processed through some simple, nonlinear function to produce the output activation $a_k$ of neuron $k$ according to

$$a_k = g(z_k) \,. \tag{2.2}$$

In this equation, $g(\cdot)$ defines the nonlinearity, and is often referred to as activation function

because it determines if and how a neuron is activated by its inputs. A widely used activation function is the *Rectified Linear Unit* (ReLU) [4], [6], which simply maps negative inputs to zero according to

$$g(z_k) = \max(0, z_k)\,. \tag{2.3}$$

Introducing these simple nonlinearities is a central concept because it allows DNNs to represent highly nonlinear functions through the interconnection of many such neurons [1]. Applied nonlinearities should be differentiable almost everywhere, to allow smooth gradient computations across the network. This property is crucial for the optimization process which is addressed in Section 2.1.1.

In NNs, neurons are typically structured in layers that subsequently process information. Suppose a DNN that models the input-output relation $y = f(\boldsymbol{x})$ for an arbitrary input vector $\boldsymbol{x}$ with $\boldsymbol{x} \in \mathbb{R}^{d_x}$. Assume this model to be a composition of $L$ layers. The mapping from input to output can be decomposed into a sequence of linear transformations, each followed by the application of an elementwise nonlinearity as defined by:

$$y = f_L \circ ... \circ f_l \circ ... \circ f_1\,, \tag{2.4}$$

where $f_l(\cdot)$ denotes the transformation applied by layer $l$. However, there are numerous different NN architectures that are more or less suitable for various tasks [57], [2].

### 2.1.1. Backpropagation

Suppose a neural network $y_n = f_\theta(\boldsymbol{x}_n)$ that produces a real valued output $y_n \in \mathbb{R}$ for each input sample $\boldsymbol{x}_n \in \mathbb{R}^{d_x}$. The subscript $\theta$ represents the set of learnable parameters of the network, i.e., the weights and biases, on which the network output is dependent. In a supervised learning setting, each input $\boldsymbol{x}_n$ corresponds to a true target $t_n$, which is known. The goal is to learn an optimal set of parameters $\theta$, such that the model minimizes the error between predicted outputs $y_n$ and true targets $t_n$. This is quantified by some error measure, often referred to as loss function. Let the loss be represented by $\mathcal{E}_\theta$, where $\mathcal{E}_\theta = 0$ defines a perfect mapping of inputs to outputs [1].

Since optimizing neural networks is a non-convex problem, learning an optimal parameter set is achieved by an iterative procedure. Within this procedure, the parameter set is repeatedly updated by performing gradient descent of the error function w.r.t. to model parameters. This approach builds on the Perceptron algorithm, originally proposed by F. Rosenblatt in 1958 [61]. Each parameter $\theta_q \in \theta$ is updated according to

$$\theta_q = \theta_q - \eta \frac{\partial \mathcal{E}_\theta}{\partial \theta_q}\,, \tag{2.5}$$

where $\eta$ defines the learning rate, hence, the size of parameter adjustments. The fundamental concept that enables efficient parameter updates is backpropagation [62], also known as error-backpropagation. It makes use of the multivariate chain rule for derivatives to compute the partial derivatives of the error with respect to each learnable parameter.

Consider $y_n = f_\theta(\boldsymbol{x}_n)$ to be a feed-forward NN, thus a model that processes information strictly one-directional from input to output. Suppose $j$ and $k$ denote indices for interconnected neurons contained in two successive intermediate layers of the neural network. Assume a completed forward pass, i.e., inputs $\boldsymbol{x}_n$ contained in some dataset of size $N$, have been propagated through the network, and the error $\mathcal{E}_\theta$ is readily given. Let $w_{jk} \in \theta$ be the parameter of interest that should be updated. By applying the chain rule of derivatives, the partial derivative of the error with respect to $w_{jk}$ is defined as

$$\frac{\partial \mathcal{E}}{\partial w_{jk}} = \frac{\partial a_k}{\partial w_{jk}} \cdot \frac{\partial \mathcal{E}}{\partial a_k}\,. \tag{2.6}$$

The reason for backpropagation being such an efficient algorithm, is that the partial derivative to each activation $a_k$, can be calculated as the sum of all 'incoming' partial derivatives from the connected neurons in the layer above [62]. The partial derivative with respect to some arbitrary activation $a_j$ is thus defined by:

$$\frac{\partial \mathcal{E}}{\partial a_j} = \sum_k \frac{\partial a_k}{\delta a_j} \cdot \frac{\partial \mathcal{E}}{\partial a_k}\,. \tag{2.7}$$

Equation 2.7 can be repeatedly applied, starting form the output of the network until the input is reached. The full process of backpropagating the gradient from the output down to the inputs, is often referred to as backward pass [62].

### 2.1.2. Convolutional Neural Networks

Traditional, densely connected, feed-forward neural networks, process fixed size input vectors, where features have to be presented in a predefined, yet arbitrary, order [59]. However, for tasks that process real world signals, like audio data or images, such unstructured NN have limitations [59], [2]. Natural signals are often a composition of a hierarchically ordered objects [2]. In the case of images, an object is composed of several parts, which in turn are formed by a composition of smaller motifs like textures and edges [46]. Additionally, objects in images can occur with all kinds of geometric variabilities, such as distortions, or variations in scale and translation. The same logic applies to audio data, e.g., speech, where sentences are composed of words that are sequences of vowels and consonants, or

music, where sound objects are composed of melodic and rhythmic structures, which in turn are a composition of sounds from several instruments. To account for the within class variabilities and the compositional structure of real world signals, the *Convolutional Neural Network* (CNN) was proposed [59], [2]. This network is specifically designed to process data in the form of multiple arrays, such as 2-dimensional images or audio spectrograms. Through providing characteristics like robustness against geometric distortions within the inputs, as well as invariance to a shift in scale or translation, CNNs achieve superior performance in various applications [4], [6], [63], [46], [14].

Typically, CNNs are structured in several stages, where the first few stages serve as feature extractor. Each stage is a combination of so called convolutional layers and pooling layers. Essentially, a convolutional layer is a composition of multiple units, also referred to as kernels, which are organized in feature maps. All units contained in one feature map are connected to a different local patch of their input, with a set of weights that is identical for all units contained in a feature map. This weight sharing allows each feature map to learn local features with invariance to translations. Hence, a convolutional layer represents a discrete convolution over its inputs. Supposing an single input map $\boldsymbol{a} \in \mathbb{R}^{H \times W}$, and a set of weights $W \in \mathbb{R}^{K \times K}$, a 2-dimensional discrete convolution is defined as

$$\boldsymbol{z}_{ij} = (\boldsymbol{a} \star W)_{ij} = \sum_{m=1}^{K} \sum_{n=1}^{K} \boldsymbol{a}_{i+m,j+n} W_{m,n} \,, \tag{2.8}$$

where $i$ and $j$ denote indices of the output feature map $\boldsymbol{z}$. For simplicity, Eq. 2.8 models the case of a single input feature map and a single output feature map. This process is depicted in Fig. 2.1. However, since CNNs typically generate multiple feature maps per layer, Eq. 2.8 has to be adapted to sum contributions of every input feature map into each output feature map [59], [2].

As in traditional neural networks, output activations of some convolutional layer are being processed with an elementwise nonlinearity. Between stages of convolutional layers, typically pooling is applied. It accounts for dimensionality reduction through coarse graining small local regions within feature maps. One example would be max-pooling, where only the highest value of each patch in a feature map is retained. This provides some degree of invariance against geometric distortions and scale. By stacking multiple of such stages together, higher layers are able to detect high level objects as composition of parts and textures that were detected at lower layers [59], [2], [46]. A key concept in CNNs is the receptive field. It describes the pattern-respond region of a feature extractor which defines the spatial region of an input sample that is being analyzed by a single unit contained in the last convolutional layer [59].

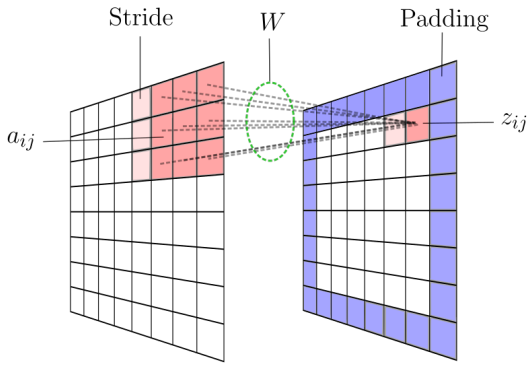



Figure 2.1.: Convolution between two feature maps of consecutive layers, supposing a single feature map per layer. Depicted is the filter set $W$, an output activation $z_{ij}$ at location $ij$ within the output feature map, and an input activation $a_{ij}$.

Important parameters in designing convolutional layers include the kernel size $K$ as defined above, the pooling size, which defines the size of the pooling kernel, as well as the stride and padding. Padding defines adding extra dimensions at the edges of feature maps and is depicted in Fig. 2.1. The stride defines the step size by which the weight kernel is shifted over the input feature maps during convolution. In the absence of padding, increasing the stride results in smaller output feature maps [4], [6], [63]. After sufficiently many convolutional stages, CNNs typically attach a classification head to the feature extractor. This classification head is typically composed of several densely connected layers that perform the decision making on the features extracted by the feature extractor.

## 2.2. Explainable Artificial Intelligence

XAI aims to reveal the opaque decision strategy of nonlinear ML methods, to provide trust in the applied systems, access their robustness, and gain insights in the data domain on which models are applied [24], [27], [30], [21], [32]. A vast amount of post-hoc explanation approaches has been proposed, which can be roughly grouped into local methods [23], [25], [26], [64], which are able to quantify the importance of input features with respect to model outputs, and global methods [65], that try to extract prototypical examples for specific output classes [30]. However, this partition is slowly being displaced, due to new research focusing on extracting richer structured explanations, such as concept-based explanations, which can be viewed as a combination of local and global XAI [35], [41], [42], [43], [66]. On the contrary to post-hoc techniques, are self-explaining models. These approaches propose to inherently design ML models in an interpretable manner [27]. However, with the aim of explaining existing models, post-hoc explanations are of need.

This chapter focuses on post-hoc explanations for classification settings, i.e., were an input

vector $\boldsymbol{x}_n$, composed of several input features according to $(x_{n,i})_{i=1}^{d_x}$, belongs to a specific target class $t_n$, and a nonlinear classifier is readily trained to predict said class by modeling the function $y_n = f(\boldsymbol{x}_n)$. The output $y_n$ is often referred to as logit score, which defines the evidence for the given class $t_n$. The subscript $n$ defines a specific sample contained some dataset with a total number of $N$ instances.

A popular approach in the realm of global XAI is activation maximization [65]. Through optimizing an input vector $\tilde{\boldsymbol{x}}$ that maximizes the activation of some neuron of a model, e.g., an output neuron, this approach is able to uncover interesting insights in how a model has encoded evidence for some class [65]. However, for a detailed examination of a models input-output behavior, such as its reactions to specific input samples for validation, the benefit of prototypical examples is rather limited [27]. In contrast, local XAI accounts for these shortcomings, yet, has limitations in extracting global explanation factors.

Local XAI constructs explanations by assigning scores to input features, indicating their relevance for the model prediction [25], [26], [23], [46], [24], [64]. Typically, these so called relevance scores are visualized within the input domain, e.g., by generating a 'heatmap' [67]. This is done by overlaying each feature of an input image with its associated relevance score. More details about the interpretability of explanations and their quality assessment are provided in Section 2.2.3.

Famous local explanation techniques include attribution-based methods like Shapley-Values [25], Integrated Gradients [26], or LRP [23], which are addressed in the next section. Another field proposes approaches that explain nonlinear classifiers with local surrogate models, which are interpretable by design [30]. One such method is the *Local Interpretable Model-agnostic Explanations* (LIME) algorithm [24]. LIME defines a model-agnostic XAI method by relying solely on inputs and corresponding outputs, i.e., it does not depend on any internals of the ML model to explain. LIME operates by measuring model outputs for synthetic data points in the neighborhood of the original sample $\boldsymbol{x}_n$, such as perturbed versions of $\boldsymbol{x}_n$. Simultaneously, a surrogate model is fitted that imitates the original model locally for a given data point. It is by design interpretable, e.g., a linear classifier [24].

### 2.2.1. Attribution-based Explanation Techniques

Attribution-based XAI comprises several methodologies. Perturbation-based approaches such as Shapley-Values [25], are model-agnostic methods that measure feature importance by occluding sets of input features and simultaneously tracking model outputs. Due to this 'indirect' computation of feature relevances, many perturbation iterations have to be performed what can result in high computational cost [27], [30].

A more direct approach are methods that make use of the gradients in DNNs. For instance, GrandietxInput [64], as its name indicates, computes relevances by multiplying input features with their associated partial derivatives coming form the output. Extending this idea, Integrated Gradients [26] starts from some reference point $\boldsymbol{x}'_n$ and integrates gradients along a straight path to the original input vector $\boldsymbol{x}_n$. This method provides a more robust setup than just evaluating the gradient of a single point, and captures feature contributions over an entire trajectory of inputs [26].

Another line of work proposes modified backpropagation processes like LRP [23]. Such approaches provide a highly efficient redistribution process of relevances, by altering the gradient computation in neural networks, and determining feature importance within a single backward pass. However, the benefits associated with these methods come with comparatively higher complexity regarding their implementation. Since LRP is an important part of this study, further elaboration on its framework is detailed in the following.

#### 2.2.1.1. Layer-wise Relevance Propagation

As backpropagation-based explanation technique, LRP is applied to neural networks. It suggests to redistribute the output, i.e., evidence for some input vector belonging to some target class, down to the inputs, layer by layer, using purposely designed propagation rules [23], [68], [69], [70]. The redistribution process is defined to ensure 'conservation', meaning that the total relevance entering a neuron at a given layer has to be passed on to the connected neurons in direction of the backward pass [23], [69]. A schematic overview of this redistribution process is shown in Fig. 2.2.

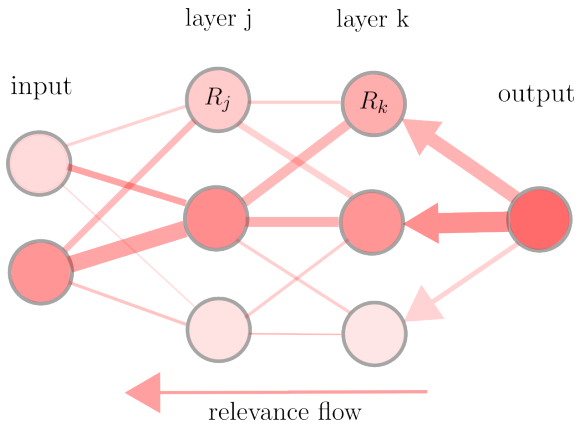



Figure 2.2.: Redistribution process of relevances in a DNN as defined by LRP.

Suppose a DNN with ReLU activations, i.e., a deep rectifier network, modeling the input-

output relation $y = f(\boldsymbol{x})$ for some classification task. This network can be viewed as a concatenation of layers as defined in Equation 2.4, where each layer is a composition of neurons of the form

$$a_k = \max\left(0, \sum\nolimits_{0,j} a_j w_{jk}\right) . \tag{2.9}$$

Indices $j$ and $k$ define neurons of two consecutive layers, and all contributions of lower layer activations $(a_j)_j$ to the activation of neuron $a_k$, are summed up by $\sum_{0,j}$. For simplification, the bias term $b_k$, stated in Eq. 2.1, is expressed by the index $j = 0$, with $a_0 = 1$ and $w_{0k} = b_k$. The most basic propagation rule to attribute relevances between two layers, is the *LRP-0* rule [23]. It redistributes relevances $R_k$ associated with neurons $a_k$ at layer $k$ onto neuron $a_j$ according to

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k \, . \tag{2.10}$$

Hence, the relevance $R_j$ is determined by a weighted sum, where relevance scores coming from neurons at layer $k$ are weighted according to the share of the contribution of neuron $a_j$ to the activation of neuron $a_k$. The denominator in Eq. 2.10 serves as normalization term to ensure the aforementioned conservation property.

**Conservation Principle.** The conservation property states that $\sum_j R_j = \sum_k R_k$, what can be verified by evaluating Eq. 2.10. In absence of neuron biasas, the stronger form of conservation $y = \sum_k R_k$ is ensured [23].

Depending on the network topology and the choice of layer, some rules are advantageous over others [69], [70]. For instance, the epsilon rule (LRP-$\epsilon$), was introduced to provide robustness against noisy gradients and is often applied to upper layers. It is defined by:

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k \, . \tag{2.11}$$

In this equation, the parameter $\epsilon$ has a filtering effect on small, rather irrelevant, relevances. On the other hand, the gamma rule (*LRP-*$\gamma$), is often preferred for lower to middle layers. It emphasizes positive contributions to provide sparser, more salient, heatmaps. The redistribution methodology is given by:

$$R_j = \sum_k \frac{a_j w_{jk} + \gamma w_{jk}^+}{\sum_{0,j} a_j w_{jk} + \gamma w_{jk}^+} R_k \, , \tag{2.12}$$

where $\gamma$ is a hyperparameter that controls the emphasis of the positive share $w_{jk}^+$ of the weight $w_{jk}$. The framework of LRP proposes to use different rules for different layers, utilizing a composite strategy [70], [69].

LRP can be developed by the framework of *Deep Taylor Decomposition* (DTD) [68]. DTD expresses some relevance $R_j$ by a first-order taylor expansion, locally at neuron $a_j$. Depending on the choice of root point for the taylor expansion, different LRP rules can be derived [68], [69]. This framework is especially useful when deriving LRP rules for new layer types, as special network topologies need careful adaption of the attribution process. LRP was already adapted to various types of neural networks including CNNs [23], [32], [71], [51], Long Short-Term Memory [72], Transformers [73], or even Graph Neural Networks [66]. Nevertheless, LRP can also be used to explain other nonlinear ML techniques through so called 'neuralization', which describes the transformation of ML models into NNs, and applying LRP there [74].

### 2.2.2. Concept-based Explanation Techniques

As stated in the introduction, concept-based XAI aims to combining global decision concepts with local visualization. Given the novelty of this research field, the array of available methods is quite small, yet elaborated in the following.

*Quatitative Testing with Concept Activation Vectors* (TCAV) tests to which extend specific, user defined concepts are represented in some given class. Thorough the predefinition of two datasets containing examples of some concept, and examples where this concept is not present, e.g. the concept 'stripes' for class zebra, a logistic regression model is trained on the activations of those datasets extracted from the original model at some layer of interest. A logistic regression model essentially classifies samples as a weighted sum of input features, by transforming its outputs into probabilities with a sigmoid nonlinearity. After training, the weights of this classification model are used as so called 'concept activation vectors'. Multiplying those with the network gradient obtained for some original sample of the associated class with respect to the layer of interest, quantifies how strong a concept is represented in this data point. Extending this idea to a global scale, TCAV calculates this concept sensitivity for all samples of one class to quantify the global contribution of some concept.

On the other hand, *Concept Relevance Propagation* (CRP) proposed by [41], extends the framework of LRP to directly filter relevances during redistribution conditioned to specific concepts. To achieve this, so called condition sets $\theta$ have to be predefined by the user that represent various concepts. However, as stated in [41], it is possible to configure these sets automatically. Through controlled masking operations in the backward pass with respect to some concept $\theta_k$, CRP highlights locally for some input sample feature relevances associated with concept $\theta_k$.

Another approach called *Multidimensional Concept Discovery* (MCD), [42], suggests to decompose the hidden feature space of neural networks, into linearly independent structures that define concepts. This is accomplished through a procedure composed of two steps: the extraction and clustering of feature vectors at some intermediate layer of a DNN, and the subsequent generation of concept subspaces by performing principal component analysis on each cluster. This approach also allows to visualize concepts locally at the inputs.

#### 2.2.2.1. Disentangled Relevant Subspace Analysis

DRSA [35], proposes a representation learning objective to extract relevant components that reassemble the reasoning structure of a DNN for some class. It proceeds by optimizing orthogonal projection matrices, virtually inserted at some layer of interest, that project activations onto relevant subspaces. A schematic overview of the subspace mapping, and filtered relevance propagation is depicted in Fig. 2.3. DRSA can be easily combined with various attribution-based XAI methods, while keeping their underlying propositions, e.g., in the case of LRP, adhering the conservation property. In particular, [35] provide derivations for the integration of Shapley-Values, Integrated Gradients, and LRP.

Suppose a DNN modeling the function $y = f(\boldsymbol{x})$ for some input vector $\boldsymbol{x}$. DRSA assumes that this input-output relation is composed of a two-step mapping, as depicted in Fig. 2.3. This two-step mapping represents a projection of some input vector $\boldsymbol{x} = (x_i)_{i=1}^{d_x}$ onto latent concepts $(\boldsymbol{h}_k)_{k=1}^{K}$, with $k$ being the index for each subspace, and further to the model output $y$. Supposing these subspaces have already been identified, a two-step attribution process is able to redistribute relevance scores to the inputs by filtering the relevance flow through some subspace. This can be defined by the subsequent redistribution steps:

$$(R_k)_{k=1}^{K} = \mathcal{E}(y, \boldsymbol{h})\,, \tag{2.13}$$

$$(R_{ik})_{i=1}^{d_x} = \mathcal{E}(R_k, \boldsymbol{x})\,. \tag{2.14}$$

Eq. 2.13 describes the attribution of the model output $y$ onto subspaces $\boldsymbol{h} = (\boldsymbol{h}_k)_{k=1}^{K}$, whereas the second propagation step, stated in Eq. 2.14, distributes concept relevance $R_k$ down to input features $(x_i)_{i=1}^{d_x}$ by keeping index $k$. In deduction, $R_{ik}$ determines the combined contribution of input feature $i$ and concept $k$ to the model prediction.

However, relevant components are initially deeply entangled between neurons of hidden layers. To extract subspaces that represent distinct concepts, [35] propose to optimize an orthogonal projection matrix $\boldsymbol{U} \in \mathbb{R}^{D \times D}$ that defines the mapping from activations onto latent concepts. This matrix is defined as a concatenation of sub-matrices $U_k$, determining the mapping onto some concept $k$, in the form of

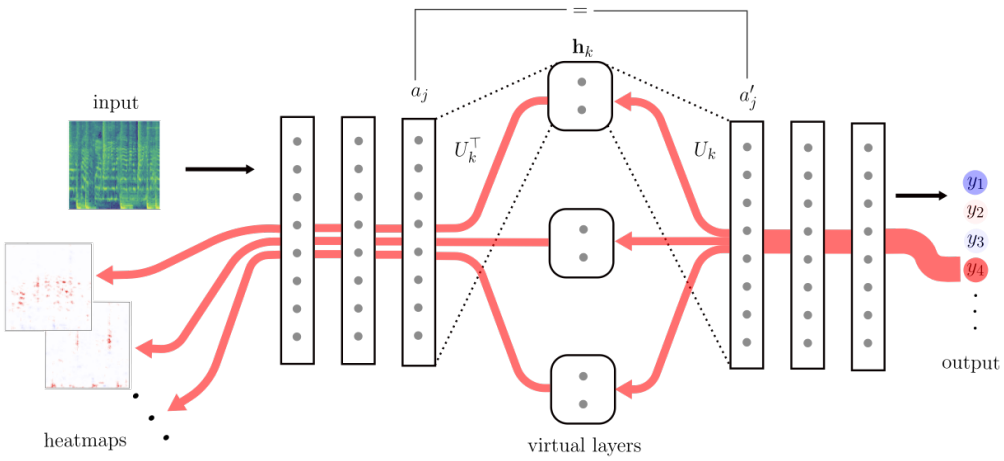



Figure 2.3.: This figure shows the virtual layers $(U_k)_k$ trained with DRSA. The black arrows depicts the forward pass of some input sample, resulting in evidence for some class (here $y_4$ for demonstration purposes). Red arrows define the relevance flow of the class evidence through relevant concepts. By keeping the relevance flow disentangled after filtering it through subspaces $\boldsymbol{h}_k$, distinct explanation components can be visualized at the inputs.

$$\boldsymbol{U} = (U_1|...|U_k|...|U_K)\,, \tag{2.15}$$

with $U_k \in \mathbb{R}^{D\times d_k}$, and $d_k$ being the dimension of subspace $\boldsymbol{h}_k$. Suppose the virtual layer is inserted at layer $j$. The linear mapping from the collection of activations $\boldsymbol{a} = (a_j)_{j=1}^D$ onto subspaces $\boldsymbol{h}_k$ is defined by

$$\boldsymbol{h}_k = U_k^\top \boldsymbol{a}\,. \tag{2.16}$$

Due to the orthogonality of $\boldsymbol{U}$, i.e., $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}_D$, activations can be easily recovered by the inverse mapping. Hence, recovering the original activation is given by

$$\boldsymbol{a}' = \sum\nolimits_{k=1}^{K} U_k U_k^\top \boldsymbol{a}\,, \tag{2.17}$$

where $\boldsymbol{a}'$ denotes the recovered activations.

Through evaluating the relevance redistribution from the consecutive layer in forward pass direction onto subspaces $\boldsymbol{h}_k$, [35] determine the relevance of concept $R_k$ according to

$$R_k = (U_k^\top \boldsymbol{a})^\top (U_k^\top \boldsymbol{c})\,. \tag{2.18}$$

The vector $\boldsymbol{c} = (c_j)_j^D$ is referred to as 'context vector', as it models the importance of some activation according to $c_j = R_j/a'_j$. With subspace relevances $R_k$ being defined, the optimization objective of DRSA proposes to find a matrix $\boldsymbol{U}$ that maximizes concept

relevances according to:

$$\begin{aligned} \underset{U}{\text{maximize}} \quad & \mathbb{M}^{q}_{k\in\{1,\dots,K\}}[\mathbb{M}^{2}_{n\in\mathcal{D}}[R^{+}_{k,n}(\boldsymbol{U})]]\,, \\ \text{subject to:} \quad & \boldsymbol{U}\boldsymbol{U}^{\top} = \boldsymbol{I}_{D}\,. \end{aligned} \tag{2.19}$$

Note that $R^{+}_{k,n}$ denotes the positive relevance of concept $k$, associated with an input sample $\boldsymbol{x}_n$ with $n \in \mathcal{D}$, and $\mathcal{D}$ denoting the dataset of samples for some class. Further, $\mathbb{M}^{p}$ denotes a generalized F-mean, with $F(t) = t^{p}$. Through setting $q < 1$, in particular $q = 0.5$ as in the original work in [35], a soft-min pooling over concept relevances is introduced that incentivizes the optimization of concepts with equal importance. On the other hand, $\mathbb{M}^{2}_{n\in\mathcal{D}}$ encourages subspaces to align with data points of high relevance.

Since Eq. 2.19 defines a non-convex problem which does not have a closed form solution, an iterative optimization approach is adapted. By starting from some random matrix $\boldsymbol{U}$, the optimization consist of an alternating application of a gradient ascent step followed by orthogonality step according to $\boldsymbol{U} \leftarrow \boldsymbol{U}(\boldsymbol{U}^{\top}\boldsymbol{U})^{-1/2}$.

### 2.2.3. Evaluating Explanations

Evaluating explanations plays a critical role in the field of XAI that encompasses two core aspects: delivering human interpretable explanations, and enhancing trust in those by quantifying their explanatory value [67], [75], [27].

#### 2.2.3.1. Interpretability of Explanations

As mentioned in the introduction, some data domains already offer an intuitively interpretable input space, where explanations can easily be delivered in a humanly comprehensive format. Common techniques in these domains include heatmapping, e.g., overlaying images with relevances, or highlighting words in text according to their associated relevance score. However, in many other fields of ML, e.g., time-series classification, visualizing explanations is not as straight forward [76], [77].

However, some approaches pave the way towards transforming explanations to increase human understandability. For example, [51] propose an extension of LRP for models trained on waveform audios, to propagate relevances 'one step further' than the input domain into time-frequency space. At this point, explanations can be visualized as heatmaps over spectrograms.

#### 2.2.3.2. Assessing Explanation Quality

Having highlighted the significance of human interpretability, evaluating explanations on a quantitative basis is of similar importance. Before gaining insights through qualitative examination, the explanatory value of the explanations has to be quantified to provide trust in those [75]. Depending on the XAI technique, quantitative evaluation methods can have fundamental impact in their design process, e.g, the choice of propagation rules in LRP [70].

A widely applied metric to quantify explanatory power is pixel-flipping [46], [75]. This method measures to what extend an extracted explanation aligns with what the models has actually encoded. In general terms, these approaches operate by occluding input features, e.g., pixels in images, from most to least relevant, while simultaneously tracking the network output for the perturbed inputs. The faster the evidence for the given target class decreases, the more truthful is the explanation [75], [78].

## 2.3. Audio Processing

Audio processing represents an essential field of research, as it tries to extract valuable meanings from sound, which is ever-present real world scenarios. Major research areas include speech recognition, audio event classification, or MIR such as MGR [8], [15], [11]. Audio processing involves challenging tasks, as audio samples may contain multiple different sounds which are often overlayed, resulting in predominantly unstructured time-series [13] [79] [48]. ML systems applied to audio problems often make use of DL approaches [7], [18], [16], [13].

### 2.3.1. Audio Representations

Sound signals naturally occur as continuous waveforms over time, often denoted as $x(t)$, where $t$ is the time at which the signals value is evaluated. They are complex waveforms, composed of a superposition of periodic sine-waves, i.e., a Fourier-series [80]. A time-continuous, periodic waveform can be formally described as

$$x(t) = A \cdot \sin\left(2\pi f t + \phi\right) \ , \tag{2.20}$$

where $A$ is the amplitude, $f$ is the frequency, i.e., the number of complete cycles of the wave per second, and $\phi$ the phase of the signal, which defines the offset of the waveforms initial cycle, typically from the reference point $t = 0$. However, to work with an audio signal in a digital environment such as in ML applications, it has to be discretized into a sequence. This is achieved by sampling values from $x(t)$ at set time points, defined by the sampling rate $f_s$. The latter is measured in Hertz (Hz), hence it defines the number of samples captured from the continuous signal per second [80]. Denote such an audio

sequence as $x[n] \in \mathbb{R}$, with $n \in \mathbb{Z}$ defining indices of the sample points.

When discretizing a continuous waveform with a sampling rate $f_s$, the highest sound frequency that can be represented by the discretized signal is defined as $f_s/2$. This is called the Nyquist-frequency (Nyquist-Shannon theorem [81]). Therefore, the higher the sampling rate, the more information is captured by the discretized signal [80]. In consequence, the sampling rate has two major impacts on audio processing tasks. Firstly, it affects the dimensionality of an audio sequence what influences processing time. Second, it determines the size of the frequency spectrum contained in the signal, what in turn can affect the precision of an ASP system.

Due to the complex nature of raw audio sequences, ASP applications often make use of other audio representations. These include spectrograms [82], mel-spectrograms [83] or *Mel-Frequency Cepstral Coefficients* (MFCC)s [84], to name a few. Those transforms decompose an audio signal into its frequency content as it varies over time, to provide a more detailed representation of the sound information. In the following, two major concepts in the realm of audio processing are detailed, in particular, the *Short-time Fourier Transform* (STFT), and the mel scale. Going forward, the term STFT will always refer to the discrete STFT.

#### 2.3.1.1. Short-Time Fourier Transform

The STFT is able to decompose a discrete, time-dependent signal into frequency components over time. Essentially, this is achieved by applying the *Discrete Fourier Transform* (DFT) to small, usually overlapping, windowed segments that get extracted from the original signal. The DFT decomposes a discretized signal of finite length into a time invariant frequency representation. It originates from the continuous-time *Fourier Transform* (FT) that provides the foundation for decomposing a complex, time-continuous signal into a Fourier Series [82], [80].

Suppose a discretized signal $x[n]$, and a finite-duration window of length $N_w \in \mathbb{N}$, with $n \in N_w$, the STFT is defined by

$$X(n, k_f) = \sum_{m=n-N_w/2}^{n+N_w/2} w[n-m] \cdot x[m] \cdot e^{-j\omega_{k_f} m} \,. \tag{2.21}$$

In this equation, $w[n-m]$ determines the window function centered at point $n$, and $e^{-j\omega_{k_f} m}$ is the complex exponential that computes the frequency component. The angular frequency $\omega_{k_f}$, is defined by $\omega_{k_f} = 2\pi \frac{k_f}{N}$, with $0 <= k_f <= \frac{N}{2}$, where $k_f$ is an index for the frequency bin evaluated by the DFT. The complex exponential is evaluated at each frequency bin,

and its relation to sinusoids is defined by the Euler's formula according to

$$e^{-j\omega_{k_f}} = \cos\omega_{k_f} + j\sin\omega_{k_f}\,. \tag{2.22}$$

Subsequently, the resulting signal in Eq. 2.22 can be decomposed into magnitude, i.e., amplitude, with

$$Mag(X(n,k_f)) = |X(n,k_f)|\,, \tag{2.23}$$

and phase, by applying

$$\Phi(n,k_f) = \tan^{-1}\frac{\Im(X(n,k_f))}{\Re(X(n,k_f))}\,. \tag{2.24}$$

In this equation, $\Im(X(n,k_f))$ and $\Re(X(n,k_f))$ denote the imaginary, and the real part of $X(n,k_f)$. The original time-domain sequence can be recovered from the complex time-frequency representation, by applying the inverse STFT [85].

Two important concepts regarding the practical application of the STFT are the window size, and the hop between each windowed frame. Whereas the hop size defines how many, and which frames are being analyzed by the STFT, the window size $N_w$ balances the time-frequency trade-off. The larger the window, the greater the frequency spectrum the signal gets decomposed into. In consequence, larger windows also include more time-increments which leads to less time-resolution. Common window functions include the Hamming window:

$$w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N_w}\right) & \text{for } 0 \le n \le N_w - 1 \\ 0 & \text{otherwise} \end{cases}\,, \tag{2.25}$$

or the rectangular window, which multiplies the original signal with 1 for $0 \le n \le N_w - 1$, and with 0 outside of the window. However, by tapering the windowed frames near their edges towards zero, the Hamming window reduces spectral leakage. The latter results from discontinuities that arise when multiplying a signal with a finite length window. It describes a distortion of the frequency spectrum, causing frequencies to spread into other frequencies [80].

In ASP applications, the 2-dimensional STFT is usually being processed as magnitude, or power spectrogram, where complex values are being transformed into real valued numbers with Eq. 2.23. In the case of power spectrograms, all amplitudes are being squared. An example of a magnitude spectrogram is visualized in the the second column of 2.4.

#### 2.3.1.2. Mel Scale

The mel scale is typically utilized to transform spectrograms into lower dimensional representations, called mel-spectrograms. These are widely used as input features for ASP applications [84], [15]. Mel-spectrograms focus on perceptually relevant features, and provide an information-rich representation while compressing the frequency axis of the STFT [83], [84]. An illustration is given in the third column in Fig. 2.4.

Before explaining the mel scale, it is important to define what pitch is. Pitch is a perceptual quantity which relates to the frequency of a tone; higher pitches correspond to higher frequencies whereas lower pitches correspond to lower frequencies. Extending this concept, the mel scale is a perceptual scale of pitches which is based on the psychology of hearing [83]. It was experimentally created to mimic the human auditory system, and relates the perceived frequency to the actual measured frequency of a tone, as humans perceive pitch on a logarithmic-like scale. They are more sensitive to changes in frequency at lower frequencies than at higher frequencies. It is important to note, that end-to-end models trained on raw waveform audios learned filter banks that correspond to the mel scale, verifying its value [15], [86].

To generate a mel-spectrogram, a set of overlapping, triangular filters is applied to a magnitude spectrogram that group frequencies according to a weighted sum into several mel bins. A general form of such triangular filters, often referred to as mel filter bank or mel bank, is depicted in Figure 2.5. A popular method to map frequencies onto the mel scale is the htk-method, proposed by [87]. It projects frequencies $f$ of a magnitude STFT onto the mel scale according to

$$Mel(f) = 2595 \cdot log_{10}(1 + \frac{f}{700}) \,, \tag{2.26}$$

Subsequently, the mel values $Mel(f)$ are grouped into a predefined number $B \in \mathbb{N}$ of evenly

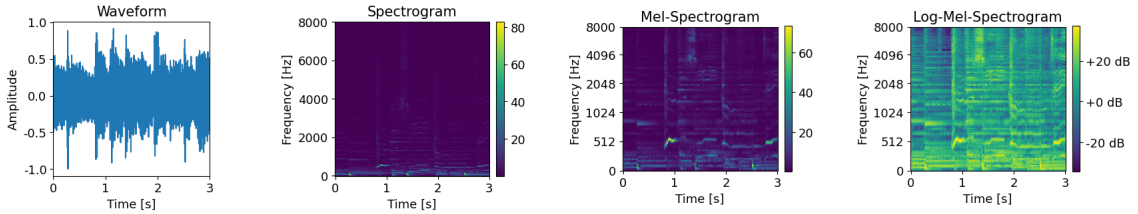



Figure 2.4.: Different audio representations obtained from a 3 second long music snippet. Form left to right: waveform audio, magnitude spectrogram, mel-spectrogram, and log-mel spectrogram with amplitudes scaled according to decibels. Note: the color bars in the standard spectrogram and mel-spectrogram depict the amplitudes.

spaced bins $m_b \in B$, which are each associated with a triangular filter. These bins define the frequency ranges from which the squared magnitudes are summed up, resulting in the final magnitudes per mel bin. The width of each triangular filter for some bin $m_b$, is given by the center frequencies of the neighboring bins as defined by $[f_{m_b-1}, f_{m_b+1}]$ [87]. Let the magnitude spectrogram be defined by $X(n, k_f)$, with $n$ denoting some time point within the spectrogram, and $k_f$ being a frequency bin associated with some frequency range. Let the weight matrix defined by some triangular filter be $W$, where entry $W(k_f, m_b)$ maps the frequencies of frequency bin $k_f$ into mel bin $m_b$. Then, the transformation is defined by:

$$M(m_b, n) = \sum_{k_f} W(k_f, m_b) \cdot X(n, k_f)\,. \tag{2.27}$$

Apart from the lower dimensional representation a mel-spectrogram provides, it incorporates another benefit for ML applications, especially for CNNs. Every sound typically consists of a fundamental frequency and harmonics, which are integer multiples of the basis frequency. When sounds are pitch shifted, i.e., their fundamental frequency is changed, the harmonics change proportionally. Hence, in standard spectrograms, sounds, e.g., melodies, that are pitch shifted, result in structures with different relative positions to each other. Since mel spectrograms are scaled logarithmically, they keep relative harmonic patterns more consistent [79].

In many cases, amplitudes of mel-spectrograms are additionally projected onto a logarithmic scale, e.g., into decibels, forming a log-mel-spectrogram. The decibel scale defines the ratio between the amplitudes of two sound sources $A_s$ and $A_{ref}$ as defined by

$$dB = 20 \log_{10}(\frac{A_s}{A_{ref}})\,, \tag{2.28}$$

though the reference value $A_{ref}$ is usually set to 1 in most practical applications. This

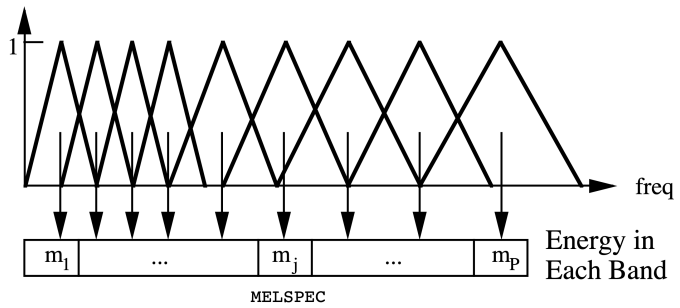



Figure 2.5.: General form of the mel filter bank obtained by the htk-method where $m_i$ defines a mel bin. This figure was adapted from [87].

logarithmic scale also imitates the human auditory system in how it perceives loudness [13]. A log-mel-spectrogram is visualized on the far right in Fig. 2.4.

### 2.3.2. Music Information Retrieval

MIR is an interdisciplinary research field that aims at the automated extraction, analysis, and retrieval of information from music [53]. Applications of MIR include MGR, or music tagging, which are particularly important to construct music recommendation systems. These systems are of increasing importance due to music consumption being mainly digital nowadays [53], [86], [15]. In consequence, MGR has been thoroughly explored in the recent years, as it is important to manage, and analyze the tremendous amount of music data on digital platforms [86]. Specific applications of MGR are structuring data, optimizing music search, or providing automatic music recommendation systems [15]. MGR can be seen as a sub-task of music audio tagging, which tries to predict multiple descriptive keywords to musical excerpts like genre, emotion, or instrument [86], [13].

Originally, MIR challenges were tackled by manually extracting specific features, e.g., tempo or pitch, from sound signals, and fitting shallow classifiers like *Support Vector Machines* (SVMs) on top for some target task. However, many MIR tasks such as genre classification and tagging are of subjective nature. This subjectivity suggests that it is challenging to completely understand the fundamental logic of the problem in order to develop suitable audio features. For this reason, ('semi') end-to-end DL approaches, which include models trained on waveform-, frequency-, and time-frequency representations, have shown promising results, as these models infer a purely data driven logic [15]. Thus, MGR problems are often tackled by employing DL methods such as CNNs, which achieve state-of-the-art performance [86], [14], [18], [13].

### 2.3.3. Explaining Music Classifiers

Approaches that interpret models trained on audio classification tasks stress that it is crucial to extract explanations on a concept level [55], [50], [52]. Additionally, many of those emphasize the importance of providing audible explanations to enhance human understandability [88], [89].

*SoundLIME* (SLIME) [54], is an extension of LIME, hence, a local, model agnostic, explanation method. It operates by segmenting an input sample uniformly in three different sequences, representing temporal, spectral and time-frequency content. An interpretable model is then fitted on synthetic samples, which are generated by perturbing features of the evaluated sequence. Consequently, SLIME is able to pinpoint important regions in the inputs guiding a models decision. On the contrary, [50] propose AudioLIME for mu-

sic classification systems, to provide a 'concept'-based explanation. In this approach, the definition of locality as defined in the LIME algorithm is changed, and defined by source separation estimates from some musical piece. Through perturbing these features, i.e., sources extracted from some audio, and tracking the model outputs, a surrogate model is able to reveal which components the model relies on. These components include different musical elements like bass lines or vocals. However, it is important to note that this approach is dependent on a source separation system which may introduce artifacts.

Another step towards concept-based explanations for music analysis is done by [52]. In this work, TCAV is applied to music classification systems. Hereby, the concept datasets are composed of musical excerpts that represent specific characteristics of music, e.g., 'bass' or 'melody with jumps'. Due to the concepts being represented by sound signals, the explanations can be easily presented as listenable audios. However, constructing such concept datasets in the domain of music analysis requires high-level domain expertise and can be time consuming until suitable concepts are found. On the account of this, [52] additionally propose an unsupervised method to extract concept datasets, yet, this approach is limited to convolutional rectifier networks. It proceeds by aggregating feature map activations from a hidden layer for various inputs samples. The activation vectors within this feature map activations define *Concept-Activation Vectors* (CAV), that each represent multiple concepts. To disentangle CAVs and generate concept-CAVs, [52] applies Non-Negative Tucker Decomposition.

# 3 | Methodological Setup

This chapter elaborates on the methodological framework used in this study.

## 3.1. Model

To tackle the audio classification tasks, a DL approach will be trained from scratch. Since CNNs have shown to achieve high performance on ASP tasks, especially in processing various types of spectrograms [13], [14], [15], a convolutional neural network architecture is chosen. Inputs to the model will be transformed into log-mel-spectrograms, providing a lower dimensional representation with a focus on perceptually relevant features.

The architecture of the employed CNN is inspired by the VGGish network, which has been successfully applied ASP and MGR tasks [18]. The model consists of a feature extractor, composed of five convolutional blocks, each comprising one 2-dimensional convolutional layer, followed by a ReLU activation and a max-pooling layer with kernel size $2 \times 2$. Building on prior research, which demonstrated that small kernels are adequate for learning meaningful representations from data, e.g., the VGG network for image classification [6], the convolutional layers employ small kernels with a shape of $3 \times 3$, zero padding and a stride of 1. On top of the feature extractor, a classification head with 3 densely connected layers is added to recognize relations between feature map activations and produce real

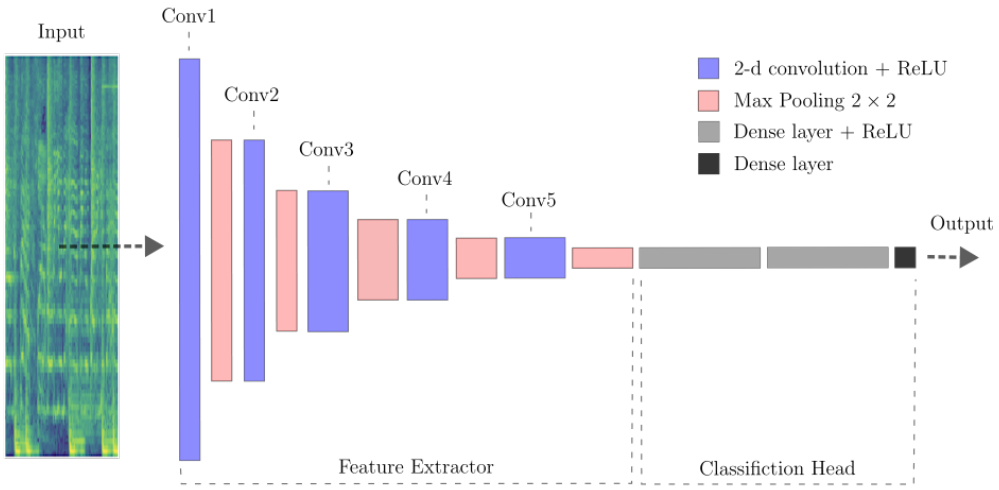



Figure 3.1.: Architectural framework of the model

valued outputs for each class. Specifications such as the number of filters per convolutional layer, or size (in terms of neurons) of the dense layers, vary depending on the performed experiment, and are provided in Chapter 4. Yet, the aforementioned architectural framework holds for all models in this study. Fig. 3.1 displays this framework schematically, with further specifications such as layer names which are important for the subsequent experiments.

## 3.2. Explanation Setup

The following sections constitute the core of this work, namely the methodology to optimize relevant subspaces, the two-step attribution of relevances to extract joint input-concept heatmaps, and the established pipeline to generate audios from explanations.

### 3.2.1. Local Attribution

To redistribute relevances, the framework of LRP will be utilized [23]. Apart from its computational efficiency, LRP has shown to be a powerful tool to gain insights into models and inputs [23], [33], [31], [71], and was previously successfully applied to ASP systems [51], [48], [90], [91], [92]. For a detailed elaboration on LRP and its benefits refer to Section 2.2.1.1.

#### 3.2.1.1. Layer-wise Relevance Propagation

Relevance propagation with LRP can be efficiently implemented through gradient computations [69]. To recover, a rearranged version of the epsilon rule that redistributes relevances onto neuron $a_j$ at some layer $j$, stated in Eq. 2.11, is defined as

$$R_j = a_j \cdot \underbrace{\sum_k \overbrace{\frac{R_k}{\epsilon + \sum_{0,j} a_j w_{jk}}}^{\text{const.}} w_{jk}}_{\text{gradient computation}} , \tag{3.1}$$

where $j$ and $k$ denote indices for neurons of two consecutive layers, and $w_{jk}$ is the weight connecting two neurons. The $\epsilon$ parameter defines the filter coefficient as described in Section 2.2.1.1. When substituting the constant term in Eq. 3.1 with some placeholder $s_k$, the gradient computation is recognizable and given as:

$$\sum\nolimits_k s_k w_{jk} = \Big[\nabla \Big(\sum\nolimits_k z_k(\boldsymbol{a}) \cdot s_k\Big)\Big]_j , \tag{3.2}$$

with

$$z_k = \sum\nolimits_j a_j \cdot w_{jk} , \tag{3.3}$$

denoting the pre activation of neuron $k$, where the bias is represented by index $a_{j=0}$ with a weight of 1.

Since most state of the art deep learning libraries, such as PyTorch [93] or Tensorflow [94], are readily equipped with an automatic differentiation engine that keeps track of network gradients, LRP rules can be efficiently implemented. This is achieved by modifying the automatic backward pass provided, e.g., the autograd engine in PyTorch. Such modifications can be implemented by utilizing so called 'hooks', which enable researchers to add custom program code, e.g., additional computations, at specific points in a program. A solid software package that offers a variety of popular LRP rules which are readily defined to redistribute relevances in models defined with PyTorch, is Zennit [95]. It implements the redistribution process by adding forward and backward hooks to individual modules (i.e., layers) of PyTorch models, to modify the gradient computations in favour of relevance distribution. For these reasons, it was selected for the attribution processes in this study.

To evaluate appropriate relevance-redistribution strategies, i.e., composite strategies for LRP, a patch flipping procedure was implemented. Details on these experiments are stated in Appendix B.1.1.

### 3.2.2. Disentangled Relevant Subspace Analysis

This section explains the extraction of relevant subspaces with DRSA.

#### 3.2.2.1. Dataset Generation

Let the CNN model the function $y = f(\boldsymbol{x}_m)$ for an input sample $\boldsymbol{x}_m$, representing a log-mel-spectrogram. The index $m$ defines instances contained in the original training set $\mathcal{M}$ for some specific class. Intermediate outputs at some convolutional layer $j$, are collections of feature map activations with shape $D \times H \times W$, i.e., the number of filters, the height of the feature map, and the width, respectively. DRSA optimizes subspaces by inserting a virtual projection layer at layer $j$, that maps activations onto latent subspaces.

To optimize subspaces, firstly, a dataset is created that is composed of activation vectors $\boldsymbol{a}_n$, and their associated context vectors $\boldsymbol{c}_n$, where $n \in \mathcal{D}$ denote indices for different data points of the training set for DRSA. One activation vector $\boldsymbol{a}_n$, for a given data point $\boldsymbol{x}_m$, is defined as the collection of activations $\boldsymbol{a}_n = (a_{n,j})_{j=1}^{D}$, sampled at a specific spatial location $p \in P$ across feature map activations obtained for $\boldsymbol{x}_m$ at some layer of interest $j$. Activations are extracted after ReLU. The total number of possible sample locations is given by $P = H \cdot W$. Vectors are then extracted by sampling $P' \in P$ random locations for each pair of activation and relevance maps. To recover, the context vectors are generated

by $c_j = R_j/a_j$ with $c_j$ being solely defined for activated neurons, i.e., $a_j \neq 0$, and $c_j = 0$ otherwise. Prior to optimization, activation and context vectors are being normalized according to

$$\hat{\boldsymbol{v}} = \frac{1}{\sqrt[4]{D}} \frac{\boldsymbol{v}}{\sqrt{\mathbb{E}_{i,j}[v_{ij}^2]}}\,, \tag{3.4}$$

where $\boldsymbol{v}$ is a placeholder for $\boldsymbol{a}$, or $\boldsymbol{c}$, respectively [35].

#### 3.2.2.2. Optimization Setup

Following the work in [35], the non-convex optimization problem, stated in Eq. 2.19, is tackled with an iterative learning procedure by starting from a random orthogonal matrix $\boldsymbol{U}$ with $\boldsymbol{U} \in \mathbb{R}^{D\times D}$. For all experiments conducted, the projection matrix $\boldsymbol{U}$ is equally grouped into $K$ 'sub-matrices' $U_k \in \mathbb{R}^{D\times d_k}$, defining the projection onto subspace $k$, with $K$ denoting the total number of subspaces. Hence, the subspace dimension $d_k$ is set to $D/K$ for all $k \in K$. A random orthogonal matrix can be sampled from the ortho-group of SciPy [96].

The optimization procedure is then performed by iteratively applying a gradient ascent step, followed by an orthogonalization step. In each iteration, gradients of the objective are computed with respect to each entry of the projection matrix $\boldsymbol{U}$. When substituting the objective of DRSA defined in Eq. 2.19 with $J(\boldsymbol{U})$, the entries of the projection matrix are updated by:

$$U_{i,j} = U_{i,j} + \frac{\partial J(\boldsymbol{U})}{\partial U_{i,j}}\,. \tag{3.5}$$

To ensure distinctiveness of subspaces, the projection matrix is orthogonalized in each optimization step following [97], [98], [35], according to

$$\boldsymbol{U} \leftarrow \boldsymbol{U}(\boldsymbol{U}^\top \boldsymbol{U})^{-1/2}\,, \tag{3.6}$$

what assures the orthogonality constraint $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_D$. To construct the inverse square-root of $\boldsymbol{U}^\top\boldsymbol{U}$, this matrix has to be decomposed. Since $\boldsymbol{U}^\top\boldsymbol{U}$ defines a real symmetric square matrix, this can be achieved by eigenvalue decomposition [99]. This matrix factorization technique is defined by decomposing some matrix $A$ according to

$$A = S\Delta S^{-1}\,, \tag{3.7}$$

where $S$ is an invertible matrix containing the eigenvectors of $A$ on its columns, and $\Delta$ denotes a diagonal matrix containing the associated eigenvalues. After decomposition, the inverse square-root of $A$ can be constructed as

$$A^{-\frac{1}{2}} = S\Delta^{-\frac{1}{2}}S^{-1}\,, \tag{3.8}$$

where $\Delta^{-\frac{1}{2}}$ defines taking the square-root of each entry on the diagonal of $\Delta$. Substituting $A$ with $U^\top U$ in Eq. 3.7, and 3.8, enables the orthogonalization step depicted in Eq. 3.6.

### 3.2.3. Two-Step Attribution

Having optimized projection matrices $U_k$ at some layer $j$ that map activations $\boldsymbol{a}_n$ associated to samples $\boldsymbol{x}_m$ onto relevant subspaces $\boldsymbol{h}_k$, a two-step attribution procedure can be implemented to extract joint input-concept heatmaps. The disentanglement of the relevance flow is performed at layer $j$ during the backward pass. Suppose the forward pass was already conducted, and relevance scores $R'_j$ are readily provided for neurons of layer $j$. After constructing the context vector $\boldsymbol{c}_n$, the mapping onto concept relevances $R_k$, associated with subspaces $\boldsymbol{h}_k$, is performed according to

$$R_k = (U_k^\top \boldsymbol{a})^\top (U_k^\top \boldsymbol{c})\,. \tag{3.9}$$

Thereafter, the conditioned relevances $R_{jk}$ for neurons $j$ at layer $j$ can be extracted with

$$R_{jk} = \frac{a_j \boldsymbol{w}_{jk}^\top \mathbf{1}}{\epsilon + \sum_j a_j \boldsymbol{w}_{jk}^\top \mathbf{1}} R_k\,, \tag{3.10}$$

what equals the LRP-$\epsilon$ rule. By retaining index $k$ of $R_{jk}$ during the subsequent backward pass, joint input-concept explanations $R_{ik}$, with $i$ defining an index for the input features of $\boldsymbol{x}_n$, are obtained. Further details on the implementation are stated in Appendix B.2.

## 3.3. Transforming Explanations into Audios

The general outline of the approach is to mask the input mel-spectrogram with the associated component heatmap, and generate a listenable audio track from the masked mel-spectrogram.

To accurately describe the inverse transformation process from heatmaps to sound signals, the notation of relevances has to be changed for this section. Since relevance scores get attributed to the input domain, the component heatmaps are in the form of mel-spectrograms. Let an input mel-spectrogram be $M(m_b, n)$, with $m_b \in B$ defining an index for the mel-bins, and $n \in N$ denoting the time bins. Let the corresponding collection of relevance scores, associated with some concept $k$, be $R_k(m_b, n)$. Initially, a mask is generated from the rectified input-concept heatmap $R_k^+(m_b, n)$, filtering $M(m_b, n)$ to only retain relevant sound elements as highlighted by the component explanation. For mask generation, a threshold $\tau_p$ was introduced which represents the $p$-th percentile of the rele-

vance scores $R_k^+(m_b, n)$, to only retain the most salient explanation factors. Subsequently, Gaussian blur is applied to smooth the resulting mask, what results in a richer sound of the audios. It follows for the relevance mask $\hat{R}_k^+(m_b, n)$:

$$\hat{R}_k^+(m_b, n) = \max_{m_b,n}(\tau_p, R_k^+(m_b, n)) * G_s \,. \tag{3.11}$$

In this equation, $G_s$ denotes a Gaussian kernel that is defined as

$$G_s(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \,, \tag{3.12}$$

where $s \in \mathbb{N}$ defines the size of the square kernel, i.e., the ranges of $x$ and $y$ which are given by $[-\frac{s-1}{2}, \frac{s-1}{2}]$. In the practical setup, the standard deviation $\sigma$ of the Gaussian kernel is set to 1 and the kernel size to 5. The threshold $\tau_p$ sit set according to the 90th-percentile. Details on the choice of $\tau_p$ are provided in Appendix B.3. Eventually, the input $M(m_b, n)$ is masked according to

$$M_k(m_b, n) = \hat{R}_k^+(m_b, n) \cdot M(m_b, n) \,, \tag{3.13}$$

only retaining features related to component $k$. Subsequently, the magnitude spectrogram can be approximated with the pseudo-inverse of the weight matrix associated with the mel-scale used in the forward transformation. Let this weight matrix be defined by $W \in \mathbb{R}^{k_f \times M_b}$ with $k_f$ defining the frequency bins (see Section 2.3.1.2 for further information). Since $W$ is usually not square, the inverse mapping from mel- to magnitude-spectrogram involves calculating a pseudo-inverse $W^\dagger$ of $W$. This can be solved by calculating a non-negative least squares solution [100], [101], which is readily implemented by various python libraries, such as SciPy [96]. For some matrix $A \in \mathbb{R}^{M \times N}$, the non-negative least squares problem is defined as

$$\begin{aligned} &\min_{\boldsymbol{x}} && \|A\boldsymbol{x}_j - \boldsymbol{b}_j\|^2 \,, \\ &\text{subject to:} && b_{jm} \geq 0 \quad \forall m \in M \,. \end{aligned} \tag{3.14}$$

Applying Eq. 3.14 to find a pseudo-inverse $A^\dagger \in \mathbb{R}^{N \times M}$, the vector $\boldsymbol{b}_j$ defines the $j$-th row of the identity matrix $I_M$, hence, $\boldsymbol{x}_j$ denotes the $m$-th row of the pseudo-inverse $A^\dagger \in \mathbb{R}^{N \times M}$. After utilizing Objective 3.14, to construct $W^\dagger$, the reconstructed magnitude spectrogram can be obtained by:

$$X_k(n, k_f) = \sum_{m_b} W^\dagger(m_b, k_f) \cdot M_k(m_b, n) \,. \tag{3.15}$$

Subsequently, $X_k(n, k_f)$ has to be transformed into a time domain signal. Since magnitude spectrograms have no phase information, usually the Griffin-Lim algorithm [85] is applied. This method alternates forward and inverse STFTs, to iteratively approximate a waveform

audio by starting from random phases and subsequently updating those. However, time domain signals recovered with this algorithm are not optimal and often incorporate a 'metallic', or 'buzzy' sound [85]. To enable a better quality of the explanation audios, the phase information $\Phi(n, k_f)$ from the original audio corresponding to data point $M(m_b, n)$, is added to the magnitude spectrogram to form a complex STFT. Such approaches were previously seen in [88], [89]. Hence, it follows for the complex spectrogram:

$$Z_k(n, k_f) = X_k(n, k_f) + \Phi(n, k_f)\,. \tag{3.16}$$

Finally, a time domain signal $x[n']$ can be recovered by applying the inverse STFT with the overlap-add method [85]. This process essentially recovers segments $x_n[n']$ of the original signal, by performing the inverse DFT to each frequency vector in $Z_k(n, k_f)$. These segments are then added together according to

$$x[n'] = \frac{\sum_n x_n[n'] w[n' - nH]}{\sum_n w^2[n' - nH]}\,, \tag{3.17}$$

where $w[n' - nH]$ denotes the window function, with $n$ being the index for each recovered frame, and $H$ defining the hop size of the analysis window. This equation is defined for $\sum_n w^2[n' - nH] \neq 0$. The inverse DFT of a finite length signal is given by

$$x_n[n'] = \sum_{k_f=0}^{N-1} Z_k(n, k_f)_n \cdot e^{i2\pi k_f n/N}\,, \tag{3.18}$$

where the index $n$ denotes one column of $Z_k(n, k_f)$. As a last step, the loudness of each audio is adjusted to correspond to the root-mean square (RMS) of the magnitudes in the original musical excerpt. After scaling, amplitude peaks above the maximum amplitude of the original audio are clamped. The RMS of a sound signal can be computed by

$$\mathrm{RMS}(\boldsymbol{x}) = \sqrt{\frac{1}{N} \sum_{n'=0}^{N-1} x[n']^2}\,. \tag{3.19}$$

The transformation process is efficiently implemented using Librosa [102]. Although this procedure seems rather complex, audios for a set of component explanations of one data point are generated in less than 1 second.

# 4 | Experiments

The experiments conducted in this work encompass an audio classification task on synthetic data and an MGR task on a music dataset, both of which are detailed in the following. This chapter provides clickable link boxes in the caption of figures, which redirect to a GitHub webpage that contains audios associated with the content displayed in the figure. These boxes show the path component which is attached to the URL of the landing page: `https://sharckhai.github.io/drsa-audio-results/`. The landing page is linked here: 🔊 .

## 4.1. Synthetic Data

### 4.1.1. Data Construction

The synthetic dataset consists of 2 classes, containing 2000 samples each. Each class is defined by 4 distinct, class specific sound objects that represent rhythmic, and melodic structures. Each generated audio sample is a superposition of up to 4 class specific audio objects, 5 random sounds and Gaussian noise with a noise strength of 0.1. Samples are generated as superpositions of periodic sine-waves with a time length of 1 second, and a synthetic sample rate of $f_s = 16000\,\text{Hz}$. Randomness is introduced by randomizing amplitude, phase, frequency, and modulation frequency from predefined ranges. Details about are stated in Table 4.1.

Table 4.1.: Specifications about class distinctive sound objects in the synthetic dataset are provided. The columns *Frequency* and *Modulation* display ranges, from which values are sampled at random for each data point. Note: this table only states some essential parameters of each object. A detailed explanation of the data generation process is depicted in Appendix D.

| | Sound object | Description | Frequency $f$ [Hz] | Modulation [Hz] |
|---|---|---|---|---|
| Class 1 | Object 1 | Modulating bass | $[100, 150]$ | $[16]$ |
| | Object 2 | Sawtooth increasing | $[500, 600]$ | $[2]$ |
| | Object 3 | Modulating sound | $[800, 1000]$ | $[3, 6]$ |
| | Object 4 | Full-wave rectified sinusoid | $[3500, 4000]$ | $[20]$ |
| Class 2 | Object 1 | Half-wave rectified sinusoid | $[100, 150]$ | $[4, 5]$ |
| | Object 2 | Sawtooth decreasing | $[500, 600]$ | $[2]$ |
| | Object 3 | Melodic structure | $[800, 1000]$ | $[16]$ |
| | Object 4 | Full-wave rectified sinusoid | $[4000, 4500]$ | $[10]$ |

In the following, a brief introduction into the data generation process is provided. The fundamental sound of each object is represented by a periodic sine-wave. Suppose some signal $x[n]$ with $n \in N$ defining the time points of the signal, and $N \in \mathbb{N}$ determining its total length. A discrete, periodic waveform is defined as

$$x[n] = a \cdot \sin(\frac{2\pi \cdot k_f \cdot n}{N} + \phi_f)\,, \tag{4.1}$$

where $\phi_f$ defines the phase of the signal, $a$ denotes the amplitude, and $k_f$ defines the frequency of the signal according to

$$f = \frac{k_f f_s}{N}\,. \tag{4.2}$$

For a time duration of 1 second, it follows $N = f_s$, and Eq. D.2 simplifies to $k_f = f$. However, the audio objects in this dataset are based on various combinations of Eq. 4.1, to form richer structured sounds, with, e.g., modulating amplitudes (all sound objects), subsequent harmonics (sound object 3 in Fig. 4.1), varying sound frequencies (sound object 3 in Fig. 4.2), time masking (sound object 1, and 3, in Fig. 4.1, and 4.2). A detailed description of the data generation process is given in Appendix D. When transformed in to log-mel-spectrograms, each data point is of shape $64 \times 64$. Details about the parameters of the employed STFT are provided at the very end of Appendix A.1.

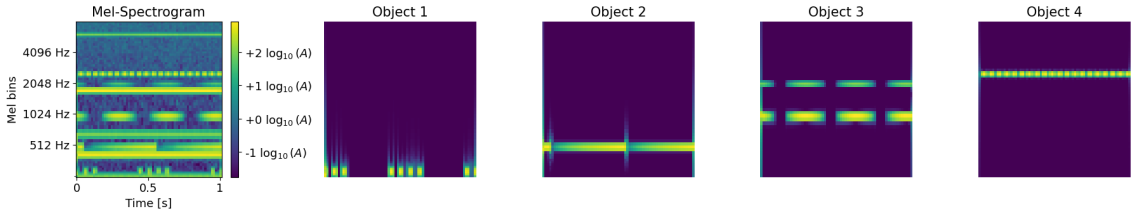



Figure 4.1.: Example of one synthetic sample contained in class 1, with each class specific audio object. Left: final sample with all 4 distinct objects, random sounds, and noise. 🔊: `synthetic/class1`

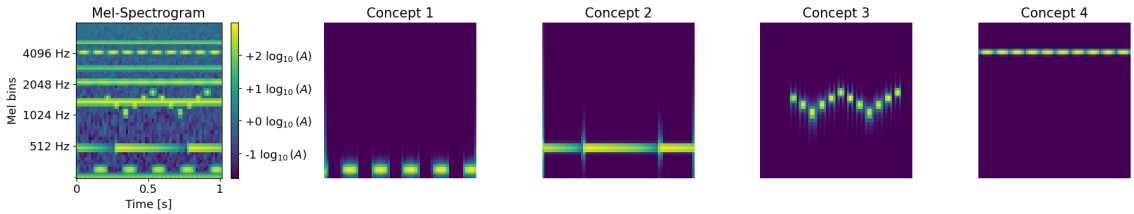



Figure 4.2.: Example of one synthetic sample contained in class 2, with each class specific audio object. Left: final sample with all 4 distinct objects, random sounds, and noise. 🔊: `synthetic/class2`

### 4.1.2. Setup

The model applied to the synthetic data employs the architectural framework stated in Section 3.1. The convolutional layers contain 8 , 8 , 16 , 16 , and 16 filters, respectively, from lower to upper layers. The first two dense layers are comprised of 20 neurons each, and the output layer contains 2 neurons, to produce logit scores for each class. The model was trained for 40 epochs and achieves a classification accuracy of 99.9% on the synthetic data. Further details about the optimization setup can be found in Appendix A

Relevances are redistributed by utilizing a composite strategy, which applies the *LRP-Flat* rule [33], to Conv1, and the LRP-$\gamma$ rule, with $\gamma$ set to 0.8, to all remaining convolutional layers. The flat rule equally redistributes the total relevance of some neuron to neurons of the lower layer. A formal definition is provided in Appendix B.1. Each dense layer employs the LRP-$\epsilon$ rule, with $\epsilon = 1 \times 10^{-7}$. For all remaining layers, i.e., activation and max-pooling layers, relevances are passed to lower layer neurons according to the gradient of each module. For LRP, the gradient of these layers already implements the intended behaviour. For instance, in the case of max-pooling layers, a winner-takes-all strategy is adapted by assigning relevances solely to the largest inputs.

To generate datasets for DRSA, activation and context vector pairs are extracted at $P = 16$ random locations within the collection of feature map activations for each input sample $\boldsymbol{x}_n$, with $n \in \mathcal{D}$. The dataset $\mathcal{D}$ defines a subset of the original training set, comprising 500 samples of one class, chosen at random. As a result, the dataset for DRSA consists of 8000 vector pairs. Optimization is performed for 5000 iterations on 3 different runs per configuration. Each run is initialized with a different (random) orthogonal matrix $\boldsymbol{U}$, and the best run is kept. Relevant subspaces are optimized at all convolutional layers throughout the network, for both classes, and a number of $K = 4$ subspaces. Training of one configuration takes about 2 minutes of a single Nvidia A100 gpu processor.

### 4.1.3. Evaluation

The extracted components for 3 different instances of class 1, optimized at Conv3, with $K = 4$, are displayed in Fig. 4.3. Presented from left to right are the input log-mel-spectrogram, the standard heatmap obtained with LRP, and the joint input-concept heatmaps. For each heatmap, the total sum of relevances is stated in its title, where index $i$ is the index for an input feature with $i \in d_x$. Furthermore, $k \in K$ defines the index for subspace. By summing up the concept heatmaps, the standard heatmap is fully recovered. This is formally described by:

$$\sum_{i=1}^{D} R_i = \sum_{k=1}^{K} \sum_{i=1}^{D} R_{i,k} \,, \tag{4.3}$$

verifying the statement in [35], that the standard heatmap can be seen as a coarse graining of the extracted sub-explanations. This may also be apparent through visual inspection of Fig. 4.3. To interpret explanation components, Fig. 4.4 recovers examples of of each sound object contained in class 1. It can be observed that the sub-explanations extracted with DRSA indeed represent specific sound audio objects. In particular, subspaces 2, 3, and 4 (specified in the indices within the heatmap titles), correspond to objects 1, 2, and 3, respectively. Subspace $k = 1$ can be seen as highlighting some higher order relations between the first 3 audio objects.

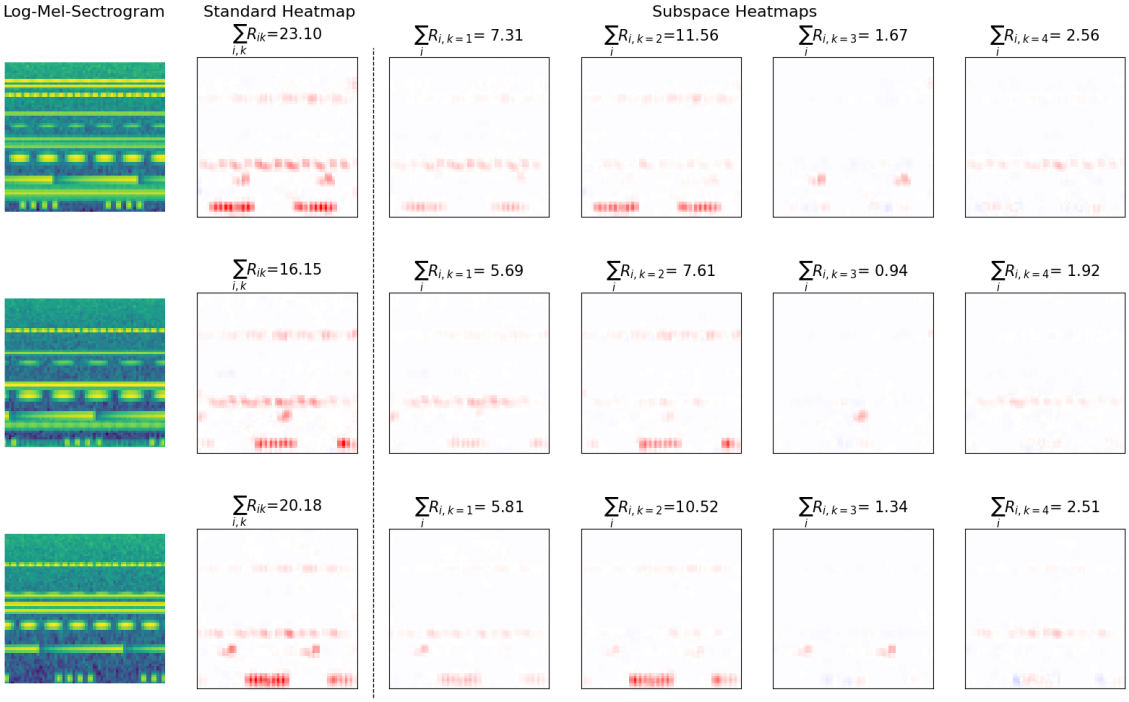



Figure 4.3.: Disentangled explanations for synthetic toy class 1 at Conv3. Depicted from left to right are the log-mel-spectrogram, the associated standard heatmap, and the component heatmaps for 3 samples of class 1. The total relevance per heatmap is defined in its title, with $R_{i,k}$ being the relevance of input feature $i$ associated with concept $k$. 🔊: `synthetic/explanations/class1`

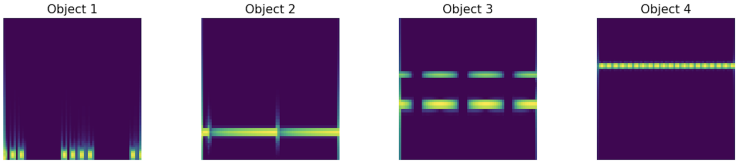



Figure 4.4.: Class specific audio objects of toy class 1. 🔊: `synthetic/class1`

It is apparent that audio object 4 on the far right in Fig. 4.4, could not be disentangled. Furthermore, it seems this object was merged into subspace $k = 2$ with audio object 1.

The primary reason for this may be that the model does not rely on this object as much, while making classification decisions. When inspecting the standard heatmaps, it is indeed the case that fewest relevance is associated with this sound signal. In consequence, it is possible that the importance associated with sound object 4 was outperformed by the merged concept represented by $k = 1$. Additionally, the similarity of sound object 4 with the random sounds, added to each sample, is comparatively high. This fact, paired with observing joint heatmap $k = 2$, leads to the assumption that the NN already merged object 4 with other objects, to enhance its classification capabilities. Results for toy class 2 can be found in Appendix C.1.

By performing an audible check of the sub-explanations obtained with DRSA (Fig. 4.3), the mapping between subspaces and audio objects can be confirmed. In addition, the value of concept-based explanations becomes evident, when comparing them to the standard explanation. Whereas the standard explanation conveys all relevant information, it is hard to tell what the model truly has encoded. The standard explanation sound like a 'cleaned' version of the original audio, without noise and random concepts. By listening to the component audios, a deeper understanding of the encoded concepts is achieved. For comparison, Fig. 4.3 depicts component heatmaps generated with a random baseline, i.e., a random projection matrix $\boldsymbol{U}$.

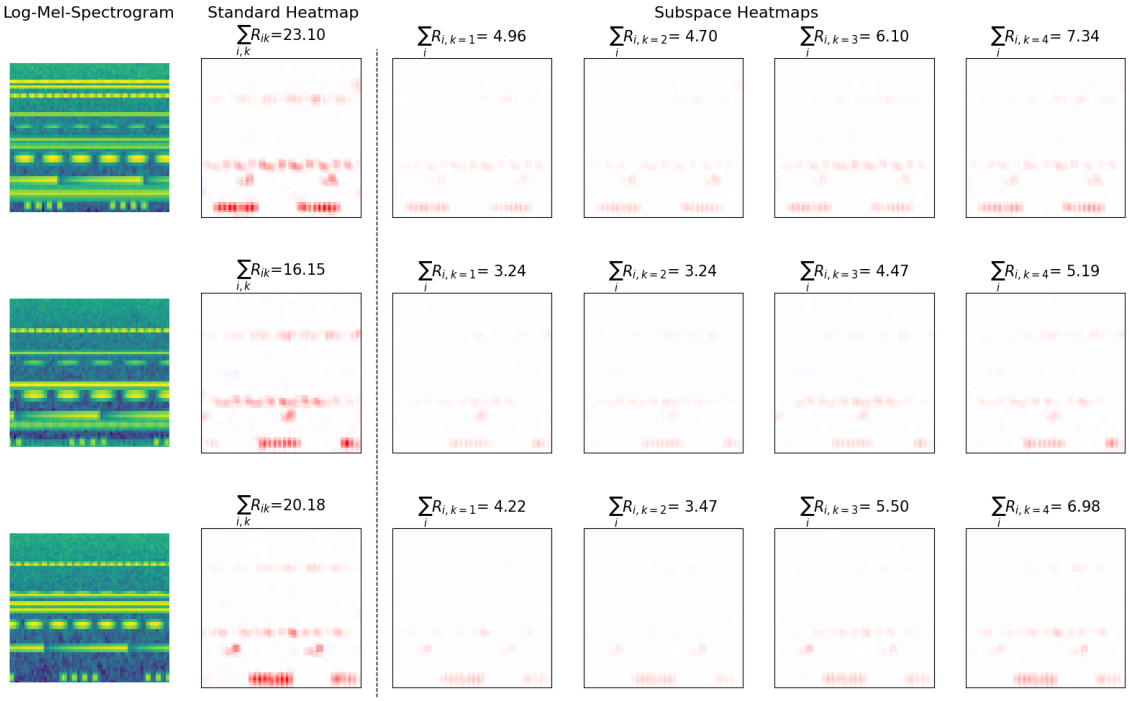



Figure 4.5.: Explanation disentanglement for samples of synthetic class 1 with $K = 4$ random subspaces at Conv3. The samples used for this experiment are the same as depicted in Fig. 4.3.

**Note.** Two aspects must be taken into account during audible verification. Firstly, due to the 'laziness' of how neural networks learn, i.e., NN often focus on prominent features of objects instead of encoding complete objects, some sounds, especially sound object 2, are not learned in their complete shape. Rather, striking parts, e.g., the regions with highest amplitude, are encoded for this object. Secondly, when comparing component audios with the exemplary sound objects provided in Fig. 4.4, a perfect alignment between subspace audios and the exemplary sound objects is very unlikely, due to the randomness in frequency and modulation.

## 4.2. Music Showcase

### 4.2.1. Setup

For the MGR task, the GTZAN dataset was selected due to its popularity [15] and feasible size. It is a widely applied dataset for ML applications in the domain of genre recognition, and is composed of 10 balanced genre classes with 1000 musical excerpts in total. Each sample incorporates a length of approximately 30 seconds. Audio tracks within this dataset were accumulated from different sources to ensure diversification in the recording quality, which include radio, CDs, and *.mp3*-files. A description of all genre classes with their corresponding number of instances, is shown in Table 4.2. Further details about data preparation and the preprocessing pipeline can be found in Appendix A.1.

Table 4.2.: Classes and samples contained in the GTZAN dataset after dataset pruning. Some sample got excluded due to distortions.

| Pop | Metal | Disco | Reggae | Blues | Classical | Rock | Hiphop | Country | Jazz |
|---|---|---|---|---|---|---|---|---|---|
| 99 | 100 | 100 | 100 | 99 | 100 | 100 | 97 | 100 | 100 |

The CNN applied to perform MGR on the GTZAN dataset, takes 3 second long music snippets transformed into log-mel-spectrograms, with a shape of $128 \times 128$, as input. The convolutional layers in the feature extractor comprise $32$ , $32$ , $64$ , $64$ , and $128$ filters in consecutive order from Conv1 to Conv5, and ensure a receptive field that spans the full input dimensions. Dense layers are composed of $128$ , $128$ , and $10$ neurons. Between dense layers, dropout is applied with a probability of 0.5, randomly zeroing out 50% of the weights during each training step. The output layer maps activations onto 10 neurons, producing real valued outputs, i.e., logit scores, for each class. Additional information about model training and evaluation are provided in Appendix A.

Following previous works [14], [13], [103], the model is evaluated on a 10-fold cross validation, and achieves an accuracy of 82.53%. State-of-the-art models achieve around 92% accuracy on the GTZAN dataset [13], [14], yet, often make use of transfer learning by utilizing feature extractors pretrained on bigger audio datasets [13], [103].

Relevances are redistributed by employing a composite strategy. The rule configuration for the feature extractor can be found in Table 4.3. The gamma-rule is applied according to Eq. 2.12, and emphasizes positive contributions to provide sparser heatmaps with a focus on more salient explanations factors. The $w^2$-rule [68], has its benefits in enhancing the direct importance of input values, by squaring the magnitude of the weights between input features and neurons of the first layer. A formal definition of this rule is provided in Appendix B.1. To each dense layers of the model, LRP-$\epsilon$ is uniformly applied with $\epsilon = 1 \times 10^{-7}$.

Table 4.3.: LRP composite for the feature extractor of the MGR model.

| Layer | LRP rule | Rule configuration |
|---|---|---|
| Conv1 | $LRP\text{-}w^2$ | - |
| Conv2 | $LRP\text{-}\gamma$ | $\gamma = 0.4$ |
| Conv3 | $LRP\text{-}\gamma$ | $\gamma = 0.4$ |
| Conv4 | $LRP\text{-}\gamma$ | $\gamma = 0.2$ |
| Conv5 | $LRP\text{-}\gamma$ | $\gamma = 0.1$ |

To generate datasets for DRSA, activation and context vectors are extracted at $P = 40$ random locations in each set of feature map activations. Since DRSA was originally showcased on models trained on image data, the practical setup was to sample 20 vectors from each collection of activations. However, although it is common that objects in images occur at various locations with different scales, music data typically spreads along time and frequency dimension of a mel-spectrogram. This allows to sample more locations, especially if less data is available, as it is the case for the GTZAN dataset. In consequence, 40 activation and context vector pairs got extracted from each collection of activations of 240 input log-mel-spectrograms of one genre class at the layer of interest, resulting in a dataset comprised of 9600 such pairs. A number of 240 input log-mel-spectrograms equals 3 snippets of each musical excerpt contained in the original training set for one class.

Optimization is performed for 5000 iterations on 3 runs per configuration with different orthogonal matrices $\boldsymbol{U}$. Relevant subspaces are optimized at all convolutional layers throughout the network, for each class, and the best run is kept. Each configuration is optimized with $K = \{2, 4, 8\}$ subspaces. Training of one configuration (with 3 runs) takes about 5 minutes on a single Nvidia A100 gpu processor.

### 4.2.2. Qualitative Evaluation

The qualitative evaluation of DRSA on the music showcase also focuses on optimization setups with $K = 4$ subspaces. These components provided visually the best disentanglement. Examples of components generated with subspaces of other layers can be found in Appendix C.2. To enhance visual inspection, the color-map of the heatmaps was changed

for this section, to emphasize smaller relevance scores, i.e., the color intensity of positive relevances increases faster.

Fig. 4.6, and 4.7 depict explanation components extracted for samples of genre class hiphop, and jazz respectively. The figures display the original data point, the standard explanation, and the sub-explanations, where the order defined in the previous section is kept. The sum of relevance scores is denoted in the title of an explanation, where index $i$ defines an input feature and $k$ denotes the corresponding subspace. The sum $\sum_{i,k} R_{i.k}$ runs over all input features, and subspaces, representing the total relevance as obtained with the standard explanation.

Again, it is apparent that the standard explanation is decomposed into several explanatory components that look spatially distinct. However, to contextualize the joint input-concept explanations, it is crucial to listen to the corresponding audios. For instance, subspaces 1 and 3, extracted for genre hiphop, represent the vocals and drums respectively. With 'drums', a combination of kick drum and snare drum is meant that usually defines the rhythm of a song, and may be classified as a typical sound element of hiphop music. Subspaces 2 and 4 seem both to highlight the kick drum at low frequencies. For the provided instances, it is not possible to distinguish those two components. It could be possible that one of these subspaces also accounts for base lines at low frequencies. For the jazz case, one may classify subspace $k = 1$ as corresponding to sounds at high frequencies. In particular, this subspace seems to represent the cymbal, which is a common instrument in jazz songs. However, this is hard to verify as this component also picks up noise, i.e., subsequent harmonics at high frequencies resulting from the main melody. Components 2, and 3, resound to the main melody, and low frequencies such as the base line respectively. Concept 4 is rather hard to interpret without specific domain knowledge. It seems to focus on some patterns in the main melody.

When comparing component explanations with the standard explanation, especially through audible inspection, the explanatory value of DRSA is again underpinned. In comparison to the synthetic data case, the standard explanation seems to be even less informative in comparison to the extracted concepts. It depicts important information, but the associated audio sounds like the original song without some melodic patterns.

Interesting concepts were also found for other genres. These include, e.g., a subspace highlighting the so called 'chop', which is a typical element in reggae music. Additionally, components for genre class metal were found that correspond to e-guitar noise and fast successive snare drums at high frequencies. Visualizations of the achieved explanation decomposition on these, and other genres, as well as links to their associated audios, can be

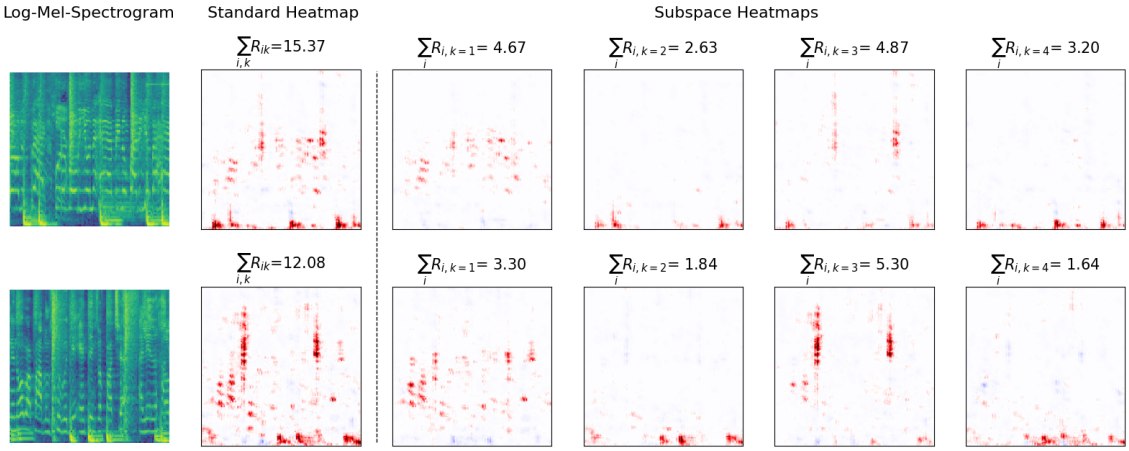



Figure 4.6.: DRSA results on genre class hiphop for $K = 4$ subspaces at Conv4. From left to right: input log-mel-spectrogram, standard heatmap, and the joint input-concept heatmaps. The value $R_{i,k}$ defines the relevance associated to input feature $i$ and concept $k$. Each row depicts the explanations for an individual instance. 🔊: `gtzan/hiphop`

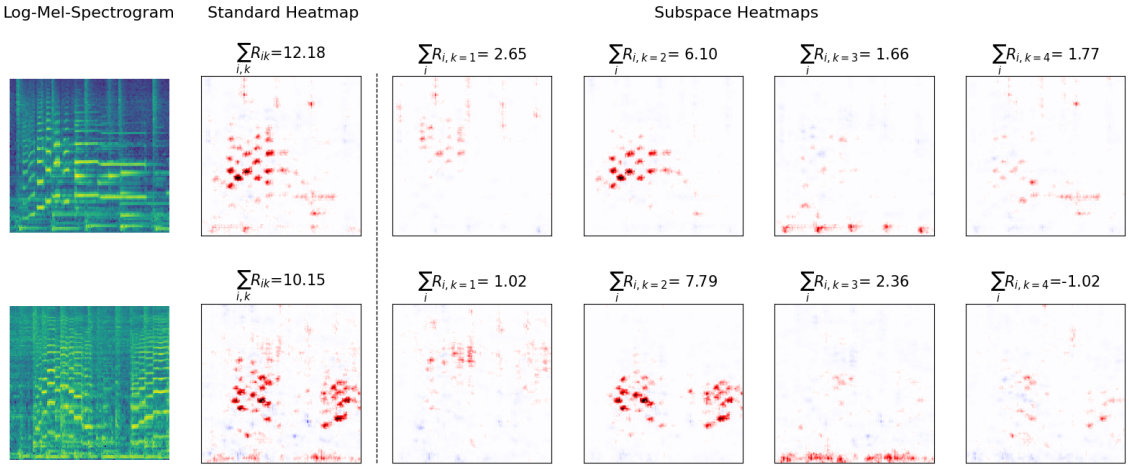



Figure 4.7.: DRSA results on genre class jazz, for $K = 4$ subspaces at Conv4. The order of images, and notation within their titles follows Fig. 4.6. 🔊: `gtzan/jazz`

found in Appendix C.2. Further, it became apparent that some subspaces across different classes highlight audio objects of similar context. Especially subspaces corresponding to kick drums or base lines, as depicted in Fig. 4.7 and 4.6, were found for every genre. On the contrary, decent vocal concepts were only extracted for genre class hiphop and blues. This observation leads to one main question: Are subspaces really distinctive between classes, or do they just correspond to specific frequency bands, e.g., low, middle, and high frequencies. To address this, further experiments have to be conducted, especially on a quantitative basis.

However, to evaluate this assumption qualitatively, Fig. 4.8 shows sub-explanations gen-

erated when propagating relevances obtained for jazz music samples through hiphop subspaces. It is apparent that the sub-explanations hardly represent distinct sound elements, when comparing the results with Fig. 4.7. The decomposition with 'false' subspaces looks rather unstructured in the sense of spatially distinct audio objects. This observation is verified by listening to the explanations. Furthermore, subspace $k = 2$, which corresponds to the kick drum of hiphop samples (see Fig. 4.6), fails to accurately disentangle the low frequencies of jazz samples. This indicates, that subspaces are indeed class specific and highlights the benefit of DRSA over dividing explanations according to frequency regions. Nevertheless, mixing up subspaces of different genres can lead to confusion and needs careful verification for multiple instances. For this reason, a quantitative approach is crucial for further analysis on the meaningfulness of explanatory components.

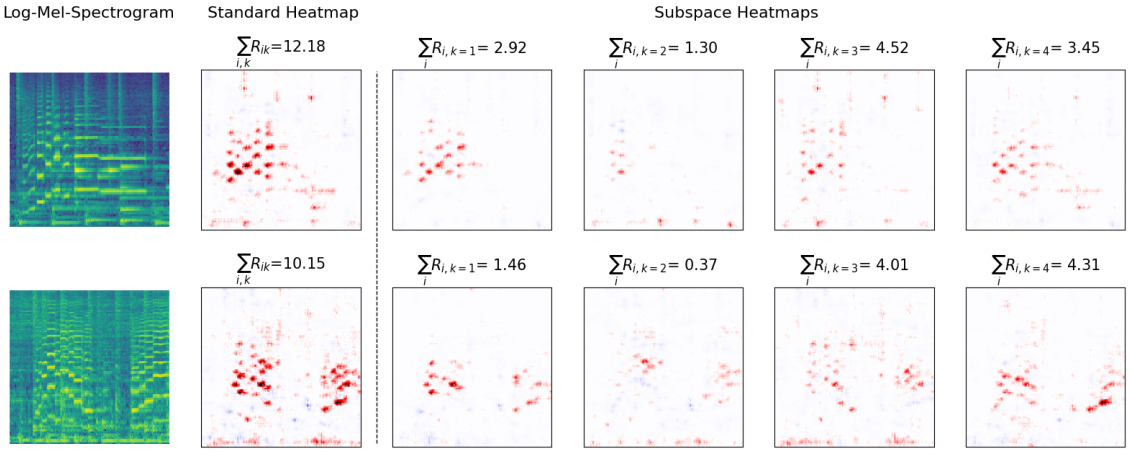



Figure 4.8.: Explanations for jazz samples disentangled with $K = 4$ subspaces at Conv4. The order of images, and notation within their titles follows Fig. 4.7.
🔊: `gtzan/special-cases`

### 4.2.3. Quantitative Evaluation

For quantitative evaluation, a patch flipping procedure was implemented in a similar fashion as in [35]. As detailed in Section 2.2.3, these approaches typically perturb patches of features in input samples from most to least relevant, while tracking network outputs of the altered inputs. The calculated *Area under the Pixel-Flipping Curve* (AUPC) score measures how faithful an explanation is. To enhance this method to evaluate the performance of explanation disentanglement, [35] propose to merge relevant patches given by each joint heatmap into one mask, before altering an input sample. Formally described, the perturbation mask $\boldsymbol{M}^\tau$ for perturbation step $\tau$, defines a unification of the masks generated for each component according to $\boldsymbol{M}^\tau = \cup_{k=1}^{K} M_k^\tau$. This means, if several sub-explanations denote the same patch as most relevant at step $\tau$, i.e., the disentanglement is not optimal, less features get flipped in the input sample. Hence, this would result in a worse score. To

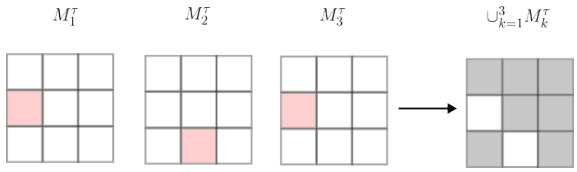



Figure 4.9.: Schematic visualization of the patch flipping procedure, implemented to address explanation disentanglement. The figure shows the generation process of a perturbation mask $\boldsymbol{M}^\tau$, obtained by unifying component masks $M_k^\tau$ perturbation step $\tau = 1$ for a setup with $K = 3$ subspaces.

avoid any confusion, an example of the mask generation is depicted in Fig. 4.9.

Suppose a model $y = f(\boldsymbol{x})$, that outputs class evidence for an input sample $\boldsymbol{x}$. Assume this input sample as $(x_i)_{i=1}^{d_x}$, where $i$ is an index for a feature contained in $\boldsymbol{x}$. The AUPC score is obtained by successively flipping all features $i$, until the complete sample $\boldsymbol{x}$ is altered. It is calculated according to

$$\mathrm{AUPC}(\boldsymbol{U}) = \mathbb{E}\left[\sum_{\tau}^{T} \omega(\tau) \left(\frac{f(\boldsymbol{x}^{(\tau-1)}) - f(\boldsymbol{x}^\tau)}{2}\right)\right], \tag{4.4}$$

where $T$ denotes the total number of perturbation steps, an altered input sample is denoted by $x^\tau$, and $\mathbb{E}$ describes an expectation over some dataset. The weight $\omega(\tau) \in (0, 1)$ defines the percentage of total features flipped at perturbation step $\tau$. To enhance efficiency, patches of size $16 \times 16$, comprising 256 input features, are being perturbed. In each perturbation step, $\tau^2$ patches are obtained from each component and unified into the final mask $\boldsymbol{M}^\tau$. Patches to perturb are extracted from most to least relevant. Their relevance score is defined as $\sum_i R_i^{(p)}$, where the superscript $p$ denotes that only relevances of features $i$ contained in patch $p$ are summed up.

Input patches are occluded through 'zeroing', i.e., by replacing each feature $x_i$ denoted by the mask $\boldsymbol{M}^\tau$ with the value 0. Although in computer vision tasks many works use generative approaches such as inpainting patches based on their surrounding pixels [104], it was observed that this introduces spurious correlations for the case of spectrogram classification. Zeroing features, assures to keep the inherent dynamics of an audio sample. Furthermore, zeroing is specifically suitable for the models applied in this study because training samples have been randomly masked by zeroing (refer to Appendix A.2 for clarification). In consequence, the applied models are invariant to patches composed of zeros.

#### 4.2.3.1. Explanation Disentanglement

DRSA is evaluated against two baselines. A random baseline, i.e., a random orthogonal matrix which is partitioned into $K$ sub-matrices defining each subspace, and an ablation

of DRSA called *Disentangled Subspace Analysis* (DSA) [35]. DSA optimizes subspaces by replacing the context vectors with the activation vectors. Hence, subspaces learned with this approach align with high activations without taking their 'context', i.e., the model response, into account. Table 4.4 states the achieved AUPC scores of each method, for an optimization setting with $K = 4$ subspaces, performed at each convolutional layer throughout the network. Further, all methods are compared with the patch-flipping score of the standard attribution, which can be defined as a setting with $K = 1$ subspace. Experiments were performed on a balanced test set composed of 1000 samples. Since the test set only contains 20 audio tracks per class, 5 distinct snippets were extracted from each of musical excerpt.

Table 4.4.: Patch-flipping scores obtained with $K = 4$ subspaces at each convolutional layer throughout the network. The best method for each setup is highlighted in bold, a lower score is better. Evaluation was performed on a balanced test set across all classes, containing 1000 samples. Scores were first averaged over instances and then over classes. The error bars depict the maximum standard error across methods. ($^\dagger$) denotes the average score across 3 random seeds.

| | Layer | | | | |
|---|---|---|---|---|---|
| | Conv1 | Conv2 | Conv3 | Conv4 | Conv5 |
| Standard attribution ($K = 1$) | 0.975 | 0.975 | 0.975 | 0.975 | 0.975 |
| Random subspaces$^\dagger$ | 0.747 | 0.702 | 0.625 | 0.561 | 0.538 |
| DSA | 0.709 | 0.613 | 0.596 | 0.540 | **0.523** |
| DRSA | **0.696** | **0.599** | **0.583** | **0.537** | 0.534 |
| *Error bars (max)* | ±0.099 | ±0.092 | ±0.079 | ±0.066 | ±0.062 |

In Table 4.4, it is apparent that DRSA achieves the best scores across convolutional layers 1-4, and is only outperformed by DSA at the last convolutional layer (layer Conv5). Furthermore, it can be observed that the disentanglement score decreases for higher layers across all methods. However, DRSA and DSA are closely followed by the random baseline, especially at higher layers. This is counter intuitive because the qualitative evaluation on the synthetic data case has shown that the subspaces extracted with DRSA produce sub-explanations of increased spatial distinctiveness, in comparison to a random baseline. However, it has to be addressed why these results are just barely represented by the patch-flipping evaluation. It is possible, that this effect occurs due to components obtained with random subspaces being distinct on a much smaller scale, e.g., at the scale of single input features. In consequence, each component mask $M_k^\tau$ could still define a different patch to flip, what results in a likewise good AUPC score.

#### 4.2.3.2. Meaningfulness of Components

As detailed in Section 4.2.2, several components across genres seem to highlight similar sounds, in particular, often respond to similar frequency bands. Since interpreting explanations in the domain of audio data is not as straight forward as in computer vision tasks, where explanations can often be easily identified by looking at heatmaps of images, the question arises if subspaces correspond to distinct audio objects across classes. This meaningfulness of components, is an important feature that has to be accessed, especially, when obtaining rather close AUPC scores with a random baseline to the ones obtained with DRSA.

To recover, in the qualitative evaluation, explanations of jazz samples were decomposed with subspaces obtained for hiphop music, and it was shown that the disentangled explanations were arguably worse compared to the components extracted with the true subspaces of the target class. However, such an approach is prone to human bias and inefficient. To address class distinctiveness of subspaces, and eliminate the possibility of DRSA dividing explanations according to frequency bands, a more exhaustive patch-flipping evaluation was performed. In this setup, explanations of one target class $c$, with $c \in C$ and $C$ being the total number of classes, i.e., $C = 10$, are decomposed $C$ times with subspaces $\boldsymbol{U}^{(c)}$ of every genre. This setup reveals similarities across classes, and enables an assessment of class distinctiveness of components.

To quantify the disentanglement of each class for cross-class comparison, the AUPC score as previously defined is not suitable. This arises from the varying scales of NN outputs for different classes. The $\Delta$AUPC score, as proposed by [35], accounts for this issue and is defined as

$$\Delta\text{AUPC}(\boldsymbol{U}) = \text{AUPC}(I_{D,K=1}) - \text{AUPC}(\boldsymbol{U})\,, \tag{4.5}$$

where $\text{AUPC}(I_{D,K=1})$ denotes the score obtained with the standard attribution. Subtracting $\text{AUPC}(\boldsymbol{U})$ from $\text{AUPC}(I_{D,K=1})$, accounts for the different scales, and enables a comparison between classes. In this case, higher scores define better disentanglement.

The cross-class $\Delta$AUPC scores are displayed in Fig. 4.10, where rows depict the attributed genre class, and columns determine the subspaces $\boldsymbol{U}^{(c)}$ used to disentangle explanations. The figure was generated with the same evaluation as used for the patch-flipping procedure in Section 4.2.3.1. In Fig. 4.10, it is apparent that the highest scores are assigned on the diagonal. Hence, the best disentanglement performance is obtained by decomposing explanations of target class $c$ with subspaces $\boldsymbol{U}^{(c)}$, for all $c \in C$. This proves the class distinctiveness of subspaces.

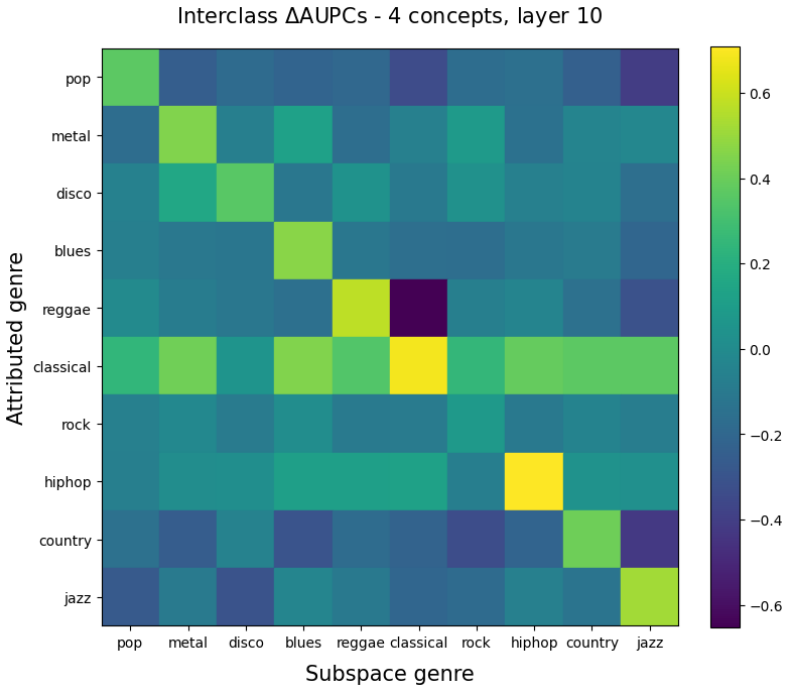



Figure 4.10.: Depicted are 10 different ΔAUPC scores for each attributed genre class. Specifically, explanations according to one target class $c$ are decomposed $C$ times with the subspaces pertaining to each class $c \in C$. The rows state which class was attributed, while the columns show which subspaces $\boldsymbol{U}^{(c)}$ were used for explanation decomposition.

Furthermore, it is observable that genre hiphop and classical, when decomposed with their true subspaces, achieve the best disentanglement across classes. Also, several other insights can be gained through inspecting the cross-class ΔAUPC plot. In particular, when inspecting target class rock (seventh row in Fig. 4.10), it is apparent that all scores are nearby, and close to zero, what denotes no noteworthy improvement against the standard explanation. Additionally, scores obtained for target class rock using its true subspaces and subspaces of genre blues, seem to be nearly identical. This reflects findings of faults contained in the GTZAN dataset, in particular, that genre class rock is prone with mislabelings, e.g., that it contains several samples of other genres like blues and country [105].

Additional interesting insights can be gained by inspecting most similar or most contradictory data classes. For instance, subspaces extracted for pop and disco music respond least to jazz music samples (last row). Reggae music achieves the worst ΔAUPC score when decomposing its explanations with subspaces extracted for classical music. On the other hand, samples of genre metal respond most to concepts extracted for blues and rock music (apart from the metal subspaces). These findings may align with human intuition on genre similarities, and therefore reflect the meaningfulness of concepts learned with DRSA. For

comparison, when using a random baseline, the cross-class $\Delta$AUPC plot would result in likewise similar colors for all entries of the matrix. However, it is also interesting that these observations are not reversible. This means that, e.g., explanations for metal music are best disentangled with blues subspaces, whereas blues samples respond most to subspaces for country music.

# 5 | Conclusion

## 5.1. Main Findings

This study demonstrated the capability of DRSA, to decompose explanations on DL models applied to audio classification tasks, into meaningful sub-explanations. A detailed pipeline was introduced to generate listenable explanations, thereby enhancing human understandability and making the results accessible to non-domain experts.

The extracted concept explanations provided superior insights into the reasoning structure of the DL approach, compared to baselines and standard explanations. This was especially underpinned through audible inspection of the explanation audios. Standard explanations mostly represented a sparser version of the input audio, with irrelevant sound objects filtered out, making a detailed interpretation of the explanations difficult. In contrast, disentangled explanations facilitated a deeper understanding of the model's reasoning structure, specifically the musical concepts it has encoded to identify a genre. These findings highlight the value of concept-based explanations for audio data, especially, for scenarios where audios are of compositional nature.

For the case of MGR, this study revealed that the main definition of 'genre' by the applied DL approaches are intrinsic relations in rhythmic and melodic structures of musical components like drums, vocals, or baselines, beyond very specific audio objects like the 'chop' in reggae music, or heavy e-guitar noise in metal music. Given the compositional nature of music, where similar instruments and vocals are common across most genres, this finding may be applicable to other MGR systems.

Moreover, this work showcased the superiority of concept-based explanations for classification models in the domain of audio data, by establishing a pipeline to produce listenable audio tracks from heatmaps on mel-spectrograms.

## 5.2. Future Work

In subsequent studies focused on explaining music content analysis systems, it would be valuable to compare explanatory concepts encoded by an MGR model trained on waveform audios with those extracted from models trained in time-frequency domain. Previous research has shown differences in the decision behaviour of models based on different audio representations, using local, attribution-based XAI [48], [49]. Furthermore, investigating

models applied to more specific music classification tasks, such as classifying 'hits', i.e., popular songs, within a single genre, may lead to interesting insights. In consequence, extracted concepts could assist in designing appropriate music generation systems. Moreover, applying DRSA to other fields of time-series analysis, like medical forecasting, or prognostic modelling, could lead to intriguing insights, and discoveries.

In addition, due to the novelty of concept-based XAI, the lack of particularly expressive evaluation metrics for decomposed explanations has to be addressed. One idea would be, to utilize metrics that include perceptual context, such as the structural-similarity index. These metrics could eventually provide more valuable results in quantifying the disentanglement and meaningfulness of explanation components.

# Bibliography


[1] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[3] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529 (2016), pp. 484–503.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386.

[5] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762.

[6] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556.

[7] Awni Hannun et al. *Deep Speech: Scaling up end-to-end speech recognition*. 2014. arXiv: 1412.5567.

[8] Hendrik Purwins et al. "Deep Learning for Audio Signal Processing". In: *CoRR* abs/1905.00078 (2019). arXiv: 1905.00078.

[9] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics*. 2019.

[10] Chung-Cheng Chiu et al. *State-of-the-art Speech Recognition With Sequence-to-Sequence Models*. 2018. arXiv: 1712.01769 [cs.CL]. URL: https://arxiv.org/abs/1712.01769.

[11] Rohit Prabhavalkar et al. *End-to-End Speech Recognition: A Survey*. 2023. arXiv: 2303.03329.

[12] Jordi Pons and Xavier Serra. *musicnn: Pre-trained convolutional neural networks for music audio tagging*. 2019. arXiv: 1909.06654.

[13] Qiuqiang Kong et al. "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition". In: *CoRR* abs/1912.10211 (2019). arXiv: 1912.10211.

[14] Caifeng Liu et al. "Bottom-up Broadcast Neural Network For Music Genre Classification". In: *CoRR* abs/1901.08928 (2019). arXiv: 1901.08928.

[15] Keunwoo Choi et al. *A Tutorial on Deep Learning for Music Information Retrieval*. 2018. arXiv: 1709.04396.

[16] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: 1609.03499 [cs.SD]. URL: https://arxiv.org/abs/1609.03499.

[17] Prafulla Dhariwal et al. *Jukebox: A Generative Model for Music*. 2020. arXiv: 2005.00341 [eess.AS]. URL: https://arxiv.org/abs/2005.00341.

[18] Shawn Hershey et al. "CNN Architectures for Large-Scale Audio Classification". In: *CoRR* abs/1609.09430 (2016). arXiv: 1609.09430.

[19] Eleni Tsalera, Andreas Papadakis, and Maria Samarakou. "Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning". In: *Journal of Sensor and Actuator Networks* 10.4 (2021). ISSN: 2224-2708. DOI: 10.3390/jsan10040072.

[20] Wei Dai et al. "Very Deep Convolutional Neural Networks for Raw Waveforms". In: *CoRR* abs/1610.00087 (2016). arXiv: 1610.00087. URL: http://arxiv.org/abs/1610.00087.

[21] David Gunning. "DARPA's explainable artificial intelligence (XAI) program". In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI '19. Marina del Ray, California: Association for Computing Machinery, 2019, p. ii. ISBN: 9781450362726.

[22] Wojciech Samek and Klaus-Robert Müller. "Towards Explainable Artificial Intelligence". In: *CoRR* abs/1909.12072 (2019). arXiv: 1909.12072.

[23] Sebastian Bach et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLOS ONE* 10.7 (July 2015), pp. 1–46. DOI: 10.1371/journal.pone.0130140.

[24] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *CoRR* abs/1602.04938 (2016). arXiv: 1602.04938.

[25] Scott Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: (2017). arXiv: 1705.07874.

[26] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: (2017). arXiv: 1703.01365.

[27] Wojciech Samek et al. "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications". In: *Proceedings of the IEEE* 109.3 (2021), pp. 247–278. DOI: 10.1109/JPROC.2021.3060483.

[28] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI". In: *CoRR* abs/1910.10045 (2019). arXiv: 1910.10045.

[29] Sajid Ali et al. "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence". In: *Information Fusion* 99 (2023), p. 101805. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2023.101805.

[30] Andreas Holzinger et al. "Explainable AI Methods - A Brief Overview". In: *xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*. Ed. by Andreas Holzinger et al. Cham: Springer International Publishing, 2022, pp. 13–38. ISBN: 978-3-031-04083-2. DOI: 10.1007/978-3-031-04083-2_2.

[31] Lorenz Linhardt, Klaus-Robert Müller, and Grégoire Montavon. "Preemptively pruning Clever-Hans strategies in deep neural networks". In: *Information Fusion* 103 (Mar. 2024), p. 102094. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2023.102094.

[32] Christopher J. Anders et al. "Analyzing ImageNet with Spectral Relevance Analysis: Towards ImageNet un-Hans'ed". In: *CoRR* abs/1912.11425 (2019). arXiv: 1912.11425.

[33] Sebastian Lapuschkin et al. "Unmasking Clever Hans Predictors and Assessing What Machines Really Learn". In: *CoRR* abs/1902.10178 (2019). arXiv: 1902.10178.

[34] Leander Weber et al. "Beyond explaining: Opportunities and challenges of XAI-based model improvement". In: *Information Fusion* 92 (2023), pp. 154–176. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2022.11.013.

[35] Pattarawat Chormai et al. "Disentangled Explanations of Neural Network Predictions by Finding Relevant Subspaces". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), pp. 1–18. ISSN: 1939-3539. DOI: 10.1109/tpami.2024.3388275.

[36] Hung Nguyen. "A Survey on Explainable Artificial Intelligence: Techniques, XAI-based Model Improvement Methods, Applications". In: (Apr. 2024). DOI: 10.13140/RG.2.2.20608.65289.

[37] Huawei Sun et al. *Utilizing Explainable AI for improving the Performance of Neural Networks*. 2022. arXiv: 2210.04686.

[38] Ribana Roscher et al. *Explainable Machine Learning for Scientific Insights and Discoveries*. 2020. DOI: 10.1109/ACCESS.2020.2976199.

[39] Frederick Klauschen et al. "Toward Explainable Artificial Intelligence for Precision Pathology". In: *Annual Review of Pathology: Mechanisms of Disease* 19 (Oct. 2023). DOI: 10.1146/annurev-pathmechdis-051222-113147.

[40] Bolei Zhou et al. "Interpretable Basis Decomposition for Visual Explanation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[41] Reduan Achtibat et al. “From attribution maps to human-understandable explanations through Concept Relevance Propagation”. In: *Nature Machine Intelligence* 5.9 (Sept. 2023), pp. 1006–1019. ISSN: 2522-5839. DOI: 10.1038/s42256-023-00711-8.

[42] Johanna Vielhaben, Stefan Blücher, and Nils Strodthoff. *Multi-dimensional concept discovery (MCD): A unifying framework with completeness guarantees*. 2023. arXiv: 2301.11911.

[43] Been Kim et al. *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)*. 2018. arXiv: 1711.11279.

[44] Amirata Ghorbani et al. *Towards Automatic Concept-based Explanations*. 2019. arXiv: 1902.03129 [stat.ML]. URL: https://arxiv.org/abs/1902.03129.

[45] Grégoire Montavon, Mikio Braun, and Klaus-Robert Müller. “Kernel analysis of deep networks”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2563–2581.

[46] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901.

[47] Bob L. Sturm. “A Simple Method to Determine if a Music Information Retrieval System is a “Horse””. In: *IEEE Transactions on Multimedia* 16.6 (2014), pp. 1636–1644. DOI: 10.1109/TMM.2014.2330697.

[48] Annika Frommholz et al. *XAI-based Comparison of Input Representations for Audio Event Classification*. 2023. arXiv: 2304.14019.

[49] Sören Becker et al. “Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals”. In: *ArXiv* abs/1807.03418 (2018).

[50] Verena Haunschmid, Ethan Manilow, and Gerhard Widmer. “audioLIME: Listenable Explanations Using Source Separation”. In: *CoRR* abs/2008.00582 (2020). arXiv: 2008.00582.

[51] “Explainable AI for time series via Virtual Inspection Layers”. In: *Pattern Recognition* 150 (2024), p. 110309. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2024.110309.

[52] Francesco Foscarin et al. *Concept-Based Techniques for "Musicologist-friendly" Explanations in a Deep Music Classifier*. 2022. arXiv: 2208.12485 [cs.SD]. URL: https://arxiv.org/abs/2208.12485.

[53] Ayush Patwari et al. “Semantically Meaningful Attributes from Cowatch Embeddings for Playlist Exploration and Expansion”. In: *International Society for Music Information Retrieval Conference*. 2020.

[54] Saumitra Mishra, Bob L. Sturm, and Simon Dixon. "Local Interpretable Model-Agnostic Explanations for Music Content Analysis". In: *International Society for Music Information Retrieval Conference*. 2017. URL: https://api.semanticscholar.org/CorpusID:795766.

[55] Saumitra Mishra et al. "Reliable Local Explanations for Machine Listening". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207444.

[56] Andreas Theissler et al. "Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions". In: *IEEE Access* 10 (2022), pp. 100700–100724. DOI: 10.1109/ACCESS.2022.3207765.

[57] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *CoRR* abs/1404.7828 (2014). arXiv: 1404.7828.

[58] Chung-Cheng Chiu et al. "State-of-the-art Speech Recognition With Sequence-to-Sequence Models". In: *CoRR* abs/1712.01769 (2017). arXiv: 1712.01769.

[59] Yann LeCun et al. "Object Recognition with Gradient-Based Learning". In: *Shape, Contour and Grouping in Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. ISBN: 978-3-540-46805-9. DOI: 10.1007/3-540-46805-6_19.

[60] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[61] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519.

[62] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536.

[63] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385.

[64] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3145–3153.

[65] Anh Nguyen et al. *Synthesizing the preferred inputs for neurons in neural networks via deep generator networks*. 2016. DOI: 10.48550/ARXIV.1605.09304.

[66] Thomas Schnake et al. "XAI for Graphs: Explaining Graph Neural Network Predictions by Identifying Relevant Walks". In: *CoRR* abs/2006.03589 (2020). arXiv: 2006.03589.

[67] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: 1312.6034.

[68] Grégoire Montavon et al. "Explaining NonLinear Classification Decisions with Deep Taylor Decomposition". In: *CoRR* abs/1512.02479 (2015). arXiv: 1512.02479.

[69] Grégoire Montavon et al. "Layer-Wise Relevance Propagation: An Overview". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Ed. by Wojciech Samek et al. Cham: Springer International Publishing, 2019, pp. 193–209. ISBN: 978-3-030-28954-6. DOI: 10.1007/978-3-030-28954-6_10.

[70] Maximilian Kohlbrenner et al. "Towards best practice in explaining neural network decisions with LRP". In: *CoRR* abs/1910.09840 (2019). arXiv: 1910.09840.

[71] Wojciech Samek et al. "Explaining the Decisions of Convolutional and Recurrent Neural Networks". In: *Mathematical Aspects of Deep Learning*. Ed. by Philipp Grohs and Gitta Kutyniok. Cambridge University Press, 2022, 229â€“266. DOI: 10.1017/9781009025096.006.

[72] Leila Arras et al. "Explaining and Interpreting LSTMs". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Ed. by Wojciech Samek et al. Vol. 11700. Lecture Notes in Computer Science. 2019, pp. 211–238. DOI: 10.1007/978-3-030-28954-6_11.

[73] Ameen Ali et al. "XAI for Transformers: Better Explanations through Conservative Propagation". In: *ICML*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 435–451.

[74] Jacob R. Kauffmann, Klaus-Robert Müller, and Grégoire Montavon. "Towards explaining anomalies: A deep Taylor decomposition of one-class models". In: *Pattern Recognit.* 101 (2020), p. 107198.

[75] Wojciech Samek et al. "Evaluating the visualization of what a Deep Neural Network has learned". In: *CoRR* abs/1509.06321 (2015). arXiv: 1509.06321.

[76] Andreas Theissler et al. "Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions". In: *IEEE Access* 10 (2022), pp. 100700–100724. DOI: 10.1109/ACCESS.2022.3207765.

[77] Ziqi Zhao et al. *Interpretation of Time-Series Deep Models: A Survey*. 2023. arXiv: 2305.14582.

[78] Stefan Blücher, Johanna Vielhaben, and Nils Strodthoff. *Decoupling Pixel Flipping and Occlusion Strategy for Consistent XAI Benchmarks*. 2024. arXiv: 2401.06654.

[79] Aurora Linh Cramer et al. “Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3852–3856. DOI: 10.1109/ICASSP.2019.8682475.

[80] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals & systems (2nd ed.)* USA: Prentice-Hall, Inc., 1996. ISBN: 0138147574.

[81] C.E. Shannon. “Communication in the Presence of Noise”. In: *Proceedings of the IRE* 37.1 (Jan. 1949), pp. 10–21. DOI: 10.1109/jrproc.1949.232969.

[82] J.B. Allen and L.R. Rabiner. “A unified approach to short-time Fourier analysis and synthesis”. In: *Proceedings of the IEEE* 65.11 (1977), pp. 1558–1564. DOI: 10.1109/PROC.1977.10770.

[83] S. S. Stevens, John E. Volkmann, and Edwin B. Newman. “A Scale for the Measurement of the Psychological Magnitude Pitch”. In: *Journal of the Acoustical Society of America* 8 (1937), pp. 185–190.

[84] S. Davis and P. Mermelstein. “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4 (1980), pp. 357–366. DOI: 10.1109/TASSP.1980.1163420.

[85] D. Griffin and Jae Lim. “Signal estimation from modified short-time Fourier transform”. In: *ICASSP ’83. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 8. 1983, pp. 804–807. DOI: 10.1109/ICASSP.1983.1172092.

[86] Keunwoo Choi, George Fazekas, and Mark Sandler. *Automatic tagging using deep convolutional neural networks*. 2016. arXiv: 1606.00298.

[87] Steve Young et al. *The HTK book*. Jan. 2002.

[88] John R. Hershey et al. “Deep clustering: Discriminative embeddings for segmentation and separation”. In: *CoRR* abs/1508.04306 (2015). arXiv: 1508.04306. URL: http://arxiv.org/abs/1508.04306.

[89] Francesco Paissan, Cem Subakan, and Mirco Ravanelli. *Posthoc Interpretation via Quantization*. 2023. arXiv: 2303.12659 [cs.AI]. URL: https://arxiv.org/abs/2303.12659.

[90] Lauréline Perotin et al. “CRNN-Based Multiple DoA Estimation Using Acoustic Intensity Features for Ambisonics Recordings”. In: *IEEE Journal of Selected Topics in Signal Processing* 13 (2019), pp. 22–33.

[91] Sören Becker et al. “Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals”. In: *CoRR* abs/1807.03418 (2018). arXiv: 1807.03418.

[92] Changhong Wang, Vincent Lostanlen, and Mathieu Lagrange. "Explainable audio Classification of Playing Techniques with Layer-wise Relevance Propagation". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095894.

[93] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703.

[94] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[95] Christopher J. Anders et al. "Software for Dataset-wide XAI: From Local Explanations to Global Insights with Zennit, CoRelAy, and ViRelAy". In: *arXiv:2106.13200* (2021).

[96] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[97] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. Vol. 26. June 2001. ISBN: 9780471405405. DOI: 10.1002/0471221317.

[98] Jan Stühmer, Richard E. Turner, and Sebastian Nowozin. *Independent Subspace Analysis for Unsupervised Learning of Disentangled Representations*. 2019. arXiv: 1909.05063.

[99] Gilbert Strang. *Linear algebra and its applications*. Belmont, CA: Thomson, Brooks/Cole, 2006.

[100] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995, pp. 36–40.

[101] Rasmus Bro and Sijmen Jong. "A Fast Non-negativity-constrained Least Squares Algorithm". In: *Journal of Chemometrics* 11 (Sept. 1997), pp. 393–401.

[102] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. Vol. 8. 2015.

[103] Daisuke Niizumi et al. *Masked Modeling Duo: Learning Representations by Encouraging Both Networks to Model the Input*. 2023. arXiv: 2210.14648.

[104] Alexandru Telea. "An Image Inpainting Technique Based on the Fast Marching Method". In: *Journal of Graphics Tools* 9.1 (2004), pp. 23–34. DOI: 10.1080/10867651.2004.10487596.

[105] Bob L. Sturm. "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use". In: *CoRR* abs/1306.1461 (2013). arXiv: 1306.1461.

[106] Jeff Hwang et al. *TorchAudio 2.1: Advancing speech recognition, self-supervised learning, and audio processing components for PyTorch*. 2023. arXiv: 2310.17864.

[107] Janne Spijkervet. *Spijkervet/torchaudio-augmentations*. 2021. DOI: 10.5281/ZENODO.4748582.

[108] Thorsteinn S. Rögnvaldsson. "A Simple Trick for Estimating the Weight Decay Parameter". In: *Neural Networks*. 1996.

# A | Model Details and Training Info

The CNNs as well as the preprocessing pipeline are implemented within the deep learning framework PyTorch [93]. In subsequent references to 'the model', it always refers to the MGR model. Details about the model applied to the synthetic data will be explicitly mentioned.

## A.1. Data Preprocessing

This section provides information about the data preprocessing of the GTZAN dataset. The preprocessing pipeline is also partly used for the synthetic data. Specifications regarding the synthetic case are provided at the very end of this section.

The model is trained to classify log-mel-spectrograms generated from 3 second long snippets, which are extracted from the original musical excerpts that have a duration of 29 -30 seconds. The audio length of 3 seconds is chosen to reduce the input dimension while avoiding information loss when generating the STFT. As stated in Section 2.3.1.1, the dimensions are adjustable through certain parameters like the window size and hop size. However, the choice of these parameters can affect either time or frequency resolution. For instance, transforming audio signals with a duration of 30 seconds into spectrograms, would either result in very high dimensional representations, or significant loss in time resolution. Additionally, slicing audios into smaller chunks enhances the dataset size.

Each excerpt contained in the dataset was preemptively downsampled from $22050\,\mathrm{Hz}$ to $16000\,\mathrm{Hz}$, which reduces the Nyquist-frequency from $11025\,\mathrm{Hz}$ to $8000\,\mathrm{Hz}$. This also facilitates more comprehensive explanations in the form of listenable audio tracks, since high frequencies are harder to discern for humans [83]. During dataset inspection, some faults were identified, e.g. parts within audios comprised of zeros. Such samples (in total 5) got excluded.

The general framework of the preprocessing pipeline involves transforming waveform audios into time-frequency domain, and applying various augmentation techniques at different stages. The implementation employs the `Torchaudio` module from PyTorch, which offers a wide variety of audio processing functionalities with enhanced efficiency [106]. Each step of the processing workflow is detailed in the following.

Table A.1.: Data augmentation during model training

| Augmentation | Probability | Randomized Parameter | Range |
|---|---|---|---|
| Gain | 0.5 | Gain value [dB] | $[-12, 3]$ |
| Pitch shift | 0.3 | Tones | $[-12, 12]$ |
| High- or lowpass | 0.4 | Cutoff frequency high [Hz] | $[200, 1400]$ |
| | | Cutoff frequency low [Hz] | $[1400, 4000]$ |
| Gaussian noise | 0.3 | Signal-to-Noise ratio | $[0.001, 0.1]$ |

Initially, a full length audio excerpt is loaded into memory. During training, a single 3 second long slice is extracted from some excerpt at a random location. Suppose the resulting audio sequence as $x[n]$ with $n \in N$, where $N$ defines the total number of discrete time steps that compose the waveform. With a given sample rate of $f_s = 16000\,\text{Hz}$ and a time length $3\,\text{s}$, the resulting audio sequence includes 48000 amplitude values. Initially, normalization by peak is performed according to

$$y[n] = \frac{x[n]}{\max_n(|x[n]|)}\,. \tag{A.1}$$

This a common choice for ASP applications, as it scales amplitudes within the range of $[-1, 1]$, while preserving the dynamic range, i.e., the ratio between loudest and softest parts. After normalization, $y[n]$ is transformed by various data augmentation techniques to improve generalization. Each augmentation is applied based on a set probability, and the parameters configuring each technique are randomly selected within predefined ranges. Augmentations and the associated parameters are depicted in Table A.1 in the same exact order as they are applied to the data samples. These transformations were implemented with the external package `Torchaudio-Augmentations` [107].

Subsequently, the waveform is transformed by the STFT into time-frequency space as outlined in Eq. 2.21, utilizing a Hamming-window (Eq. 2.25). The window spans 800 time steps, which corresponds to a time duration of 50 ms. With a hop size of 360 time points (which equals 22.5 ms), the resulting complex spectrogram $X(n', k_f)$, with $n' \in N'$ and $k_f \in K_f$, is composed of $N' = 128$ time bins, and $K_f = 399$ frequency bins. At this point, time stretching is applied with a stretching factor randomly sampled within the range $[0.8, 1.2]$. Thereafter, the stretched representation is either zero padded up to $N' = 128$ time bins, in case of a speed up, or frequency bins are discarded after the 128th time bin. Eventually, the frequency bins of the associated magnitude spectrogram $|X(n', k_f)|$ are projected onto the mel scale according to Eq. 2.27, and grouped into $M_b = 128$ mel bins. The result is a mel-spectrogram $M(m_b, n')$ with $m_b \in M_b$, comprising a final input shape of $128 \times 128$. Further, amplitudes are mapped onto the $\log_{10}$ scale with

$$M(m_b, n') = \log_{10}(M(m_b, n') + 1 \times 10^{-7})\,. \tag{A.2}$$

Values below a $\log_{10}$-amplitude of $-4$ are clamped, which is chosen due to very few samples incorporating distorted values. With this pipeline, i.e., normalizing waveforms by peak, and taking the $\log_{10}$, ensures all samples lie within a range of about $[-4, 3]$ by leaving their dynamics unchanged. In particular, the mean amplitude averaged across all instances in the dataset is $\mu_A = -0,378$, with a standard deviation of $\sigma_A = 0,326$. Finally, time and frequency masking is applied by zeroing out one segment per dimension. Masks are generated at random with a maximum width of 20 bins per dimension. All augmentations discussed are applied solely during the training phase. In the evaluation phase, it is crucial to avoid introducing randomness to guarantee consistent and reliable results. Hence, instead of of extracting one random snippet per sample, each full-length audio excerpt is divided into 10 segments of 3 seconds each, ensuring that all validation data is utilized in each epoch.

**Note.** Spectrograms generated by most python libraries are 'upside down', i.e. the frequency bins corresponding to the lowest frequencies are located at the top of the two-dimensional representation. When visualizing spectrograms, the frequency axis is usually flipped. This applies also to this work.

**Synthetic data.** Synthetic samples are preprocessed with the same pipeline used for the music showcase, yet, without the waveform augmentations state in Table A.1. Furthermore, synthetic samples are transformed into a log-mel-spectrogram by employing a window size of 400 time points, a hop size of 240, and 64 mel bins. The resulting data points are of shape $64 \times 67$, where the last 3 time bins get discarded to obtain the square shape $64 \times 64$.

## A.2. Model Optimization

The model was trained to optimize the categorical cross-entropy, which is a common choice for multi-class classification problems [60]. Let the CNN represent the function $y_{n,c} = f_\theta(\boldsymbol{x}_{n,c})$, with $\theta$ denoting the learnable parameters of the network, and $\boldsymbol{x}_{n,c}$ denotes an input sample out of the total number of instances $N$, contained in the training set of class $c$. The cross-entropy loss can be computed with

$$Err(\theta) = -\frac{1}{N}\sum_{c=1}^{C}\sum_{n=1}^{N} y_{n,c} \cdot \log h_c(f_\theta(x_n))\,, \tag{A.3}$$

where the sum $\sum_{c=1}^{C}$ runs over all classes $C$, and $h_c(\cdot)$ defines the network output for class $c$ according to a softmax activation over the output neurons. The latter function is defined

as

$$h_c(f_\theta(x_n)) = \frac{\exp(f_\theta(x_n)_c)}{\sum_{c'=1}^{C} \exp(f_\theta(x_n)_{c'})}\,, \tag{A.4}$$

where $f_\theta(x_n)_c$ defines the network output for some class $c$. Furthermore, L2-regularization is applied during optimization, to penalize parameters $\theta$ for growing too large [108]. This is achieved by adding the L2-norm to the total error, as defined by:

$$Err(\theta) = Err(\theta) + \lambda\|\theta\|_2^2\,. \tag{A.5}$$

In Eq. A.5, $\lambda$ denotes the weight decay parameter which is set to $1 \times 10^{-4}$ for model training. Training is performed by utilizing the ADAM optimizer provided by PyTorch, with a learning rate of $4 \times 10^{-4}$.

**Synthetic data case.** The model applied to the synthetic data is trained with a learning rate of $1 \times 10^{-3}$, without weight decay. Everything else aligns with the optimization setup utilized for the music showcase.

## A.3. Model Evaluation

The training curves of the MGR model are shown in Fig. A.1. It is observable that the validation loss slightly increases after epoch 500. However, this is acceptable due to the validation accuracy still improving. This increase in the loss is most likely a result of the model getting more confident about its predictions, i.e. producing higher outputs which can also lead to higher losses. Furthermore, Fig. A.2 displays a confusion matrix depicting the averaged classification accuracy per class on its diagonal. Additionally, as mentioned in the qualitative evaluation in Section 4.2.2, the rather low classification accuracy of genre rock aligns with the findings by [105], that class rock is prone with mislabelings. Furthermore, some similarities with the cross-class $\Delta$AUPC plot in Section 4.2.3 are apparent, reflecting that relevant subspaces align with what the model has actually learned.

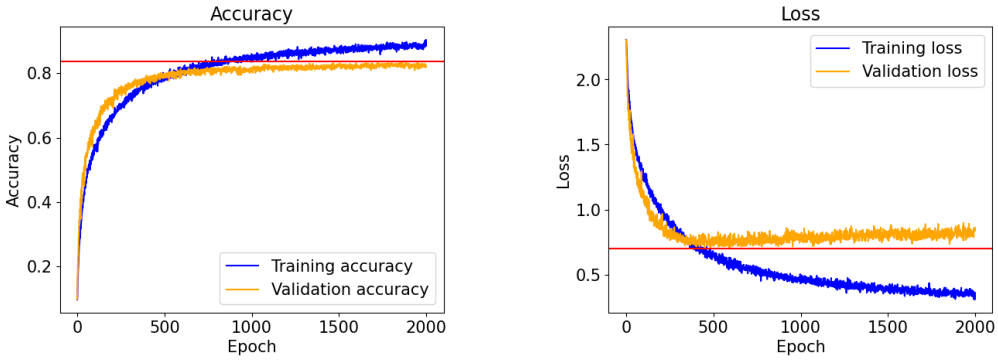



Figure A.1.: Training curves during model optimization on the music showcase, averaged over 10 folds. Left: training and validation accuracy. Right: training and validation loss.

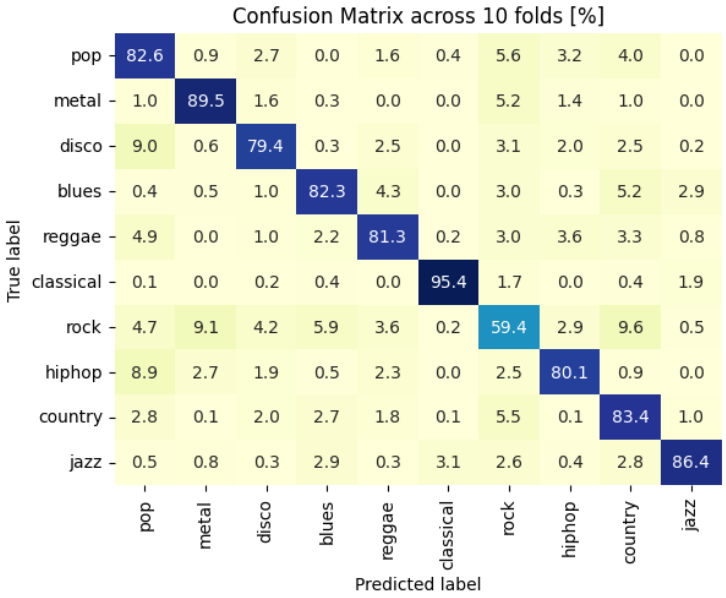



Figure A.2.: Confusion matrix depicting the class-wise accuracy on its diagonal, averaged across 10 folds.

# B | Implementation Details

The code for all experiments performed in this work is provided at:
`https://github.com/sharckhai/drsa-audio`

## B.1. Layer-wise Relevance Propagation

**LRP-$w^2$ rule** [68]**.** Let $R_j$, and $R_k$ define the relevance scores of neurons located in two consecutive layers. The $w^2$-rule is given by

$$R_j = \sum_k \frac{w_{jk}}{\sum_j w_{jk}} R_k \,, \tag{B.1}$$

with $w_{jk}$ being the weight connecting neurons $a_j$, and $a_k$. The sums $\sum_j$ and $\sum_k$ run over all neurons contained in layer $j$ and $k$ respectively.

**LRP-*Flat* rule** [33]**.** The flat-rule redistributes all relevance equally onto neurons of lower layers to provide providing smooth heatmaps. It is defined by

$$R_j = \sum_k \frac{1}{\sum_j 1} R_k \,. \tag{B.2}$$

**Practical consideration.** The redistribution process is implemented within the function `compute_relevances` in `attribute.py`, which is located in `cxai/xai`. To allow relevance redistribution for batches containing instances of all classes, an output modifier function was implemented (`lrp_output_modifier` in `attribute.py`). This functionality is particularly useful for patch flipping evaluations, as heatmaps for samples of different classes can be generated simultaneously.

### B.1.1. Evaluating Local Explanations

To access the explanation performance of LRP to accurately design LRP-composites for the employed models, a patch-flipping procedure was utilized. It is essentially similar to the algorithm defined in Section 4.2.3, yet, with the number of subspaces $K$ set to 1. A standard heatmaps is then defined as the component heatmap $k = 1$, and the unification of perturbation masks is omitted by $\boldsymbol{M}^\tau = \sum_{k=1}^{1} M_k^\tau$. Results of different patch-flipping evaluations for the model on the music showcase are depicted in Fig. B.1.

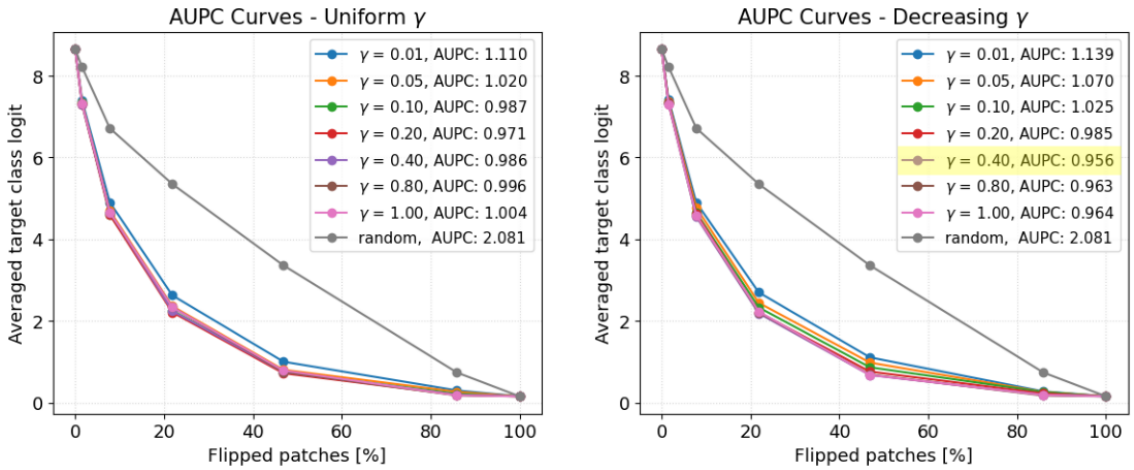



Figure B.1.: Patch-flipping evaluation of LRP on the music showcase. Left: uniform LRP-$\gamma$ across all convolutional layers. Right: decreasing $\gamma$ for subsequent convolutional layers, where the maximum $\gamma$-value is displayed. Both configurations apply LRP-$w^2$ to Conv1, and LRP-$\epsilon$ with $\epsilon = 1 \times 10^{-7}$ to all dense layers. For the random baseline, patches are flipped at randomly instead of taking their total relevance as measurement. Evaluation was performed on a balanced dataset comprising 200 log-mel-spectrograms contained in the validation set. The best score is achieved for the decreasing $\gamma$ configuration (highlighted in yellow), with an initial value of 0.4 (lower score is better). Details on the exact rule-setup are provided in Table 4.3.

## B.2. Two-step Attribution and Disentangled Explanations

### B.2.1. Optimization of Subspaces

The optimization algorithm of DRSA is implemented in `drsa.py`. Learning curves for $K = 4$ subspaces, optimized at layer Conv4 are displayed in Fig. B.2.

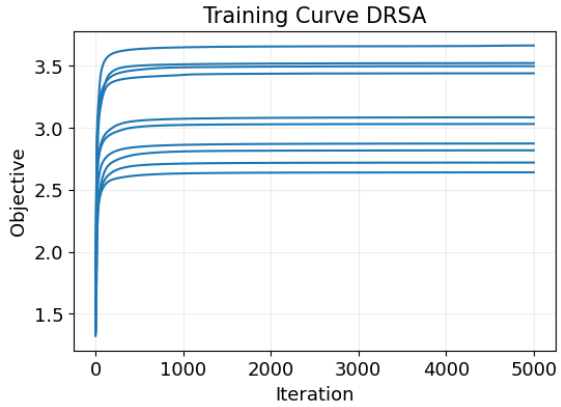



Figure B.2.: Learning curves of DRSA for $K = 4$ subspaces, optimized at layer Conv4.

### B.2.2. Two-step Attribution

The redistribution process is implemented in the class `Explainer` in `explain.py`, which is located in the sub-directory `cxai/xai/`. It starts by repeating each data point in the batch $K+1$ times, to be able to extract the standard heatmap, and all joint heatmaps within a single forward and backward pass. Suppose a data point as $x_n$. After repetition, all instances of one individual data point are defined by $x_{n,m}$ with $m \in K+1$. During the backward pass, concept relevances $(R_k)_{k=1}^{K}$ are computed for each sample in the batch, according to Eq. 3.9. At this stage, the collection of subspace relevances $(R_k)_{k=1}^{K}$, associated with the duplicates $(x_{n,m})_{m=1}^{K}$ are masked, such that solely the relevance score $R_{k=m}$ remains for the subsequent backward pass. The first sample of each mini-batch, i.e. the original instance $x_{n,m=0}$, keeps the relevances of all subspaces, i.e. sums concept relevances according to $\sum_{k=1}^{K} R_k$. This enables the extraction of the standard heatmap for instance $x_{n,m=0}$. Subsequently, subspace relevances can be attributed onto the preceding layer $j$ with Eq. 3.10. Form this point, the normal redistribution process as defined within the LRP composite can be performed, to attribute relevances onto input features.

## B.3. Audios from Explanations

The transformation pipeline is implemented in the class `Mel2Audio` in `audiogen.py`. This class has to be initialized with a data point (log-mel-spectrogram), the component heatmaps, and the path to the original audio sample to extract the phase information. The threshold $\tau_p$, mentioned in Section 3.3, is calculated for each sample according to the 90th-percentile. This value showed empirically the best results. However, a histogram plot averaged over rectified, normalized component heatmaps with $K = 4$ subspaces of 600 mel-spectrograms (i.e. 2400 component heatmaps), is displayed in Fig. B.3. It depicts the 90th-percentile of relevance scores, as well as the summation of mean and standard deviation.

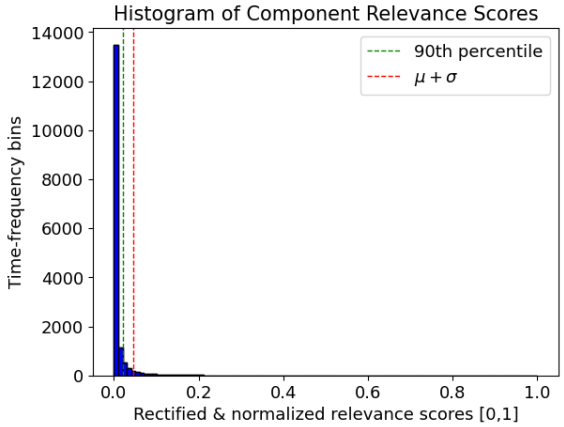



Figure B.3.: Histogram plot of the averaged relevance distribution across 2400 component heatmaps, generated from a dataset comprising 600 log-mel-spectrograms.

# C | Qualitative Evaluation Supplement

## C.1. Synthetic Data

Fig. C.1 displays explanations extracted for synthetic toy class 2. Exemplary audio objects of class 2 are provided within Fig. C.2. In each title, the index $k$ defines to which subspace each heatmap corresponds. It is apparent that especially audio objects 3, and 4 (in Fig. C.2) are decently disentangled, and represented by subspaces $k = 2$, and $k = 3$ respectively. Fig. C.3 shows explanations extracted at different layers throughout the network. We can see that for lower layers, no noticeable disentanglement can be achieved, and for the ultimate convolutional layer, the components collapse into one main explanation.

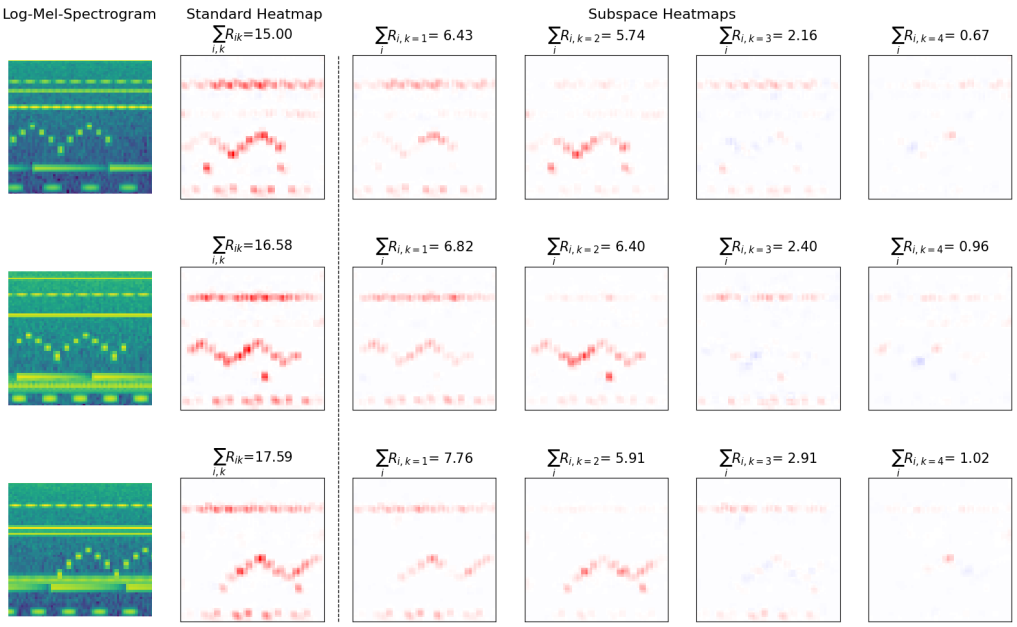



Figure C.1.: Multiple examples of the explanation disentanglement achieved with DRSA on toy class 2 at Conv3. Order of images, and Notation within their titles follows Fig. 4.3. 🔊: `synthetic/explanations/class2`

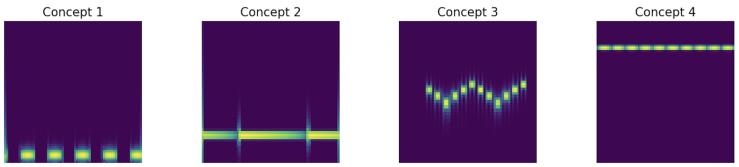



Figure C.2.: Class specific audio objects of synthetic class 2. 🔊: `synthetic/class2`

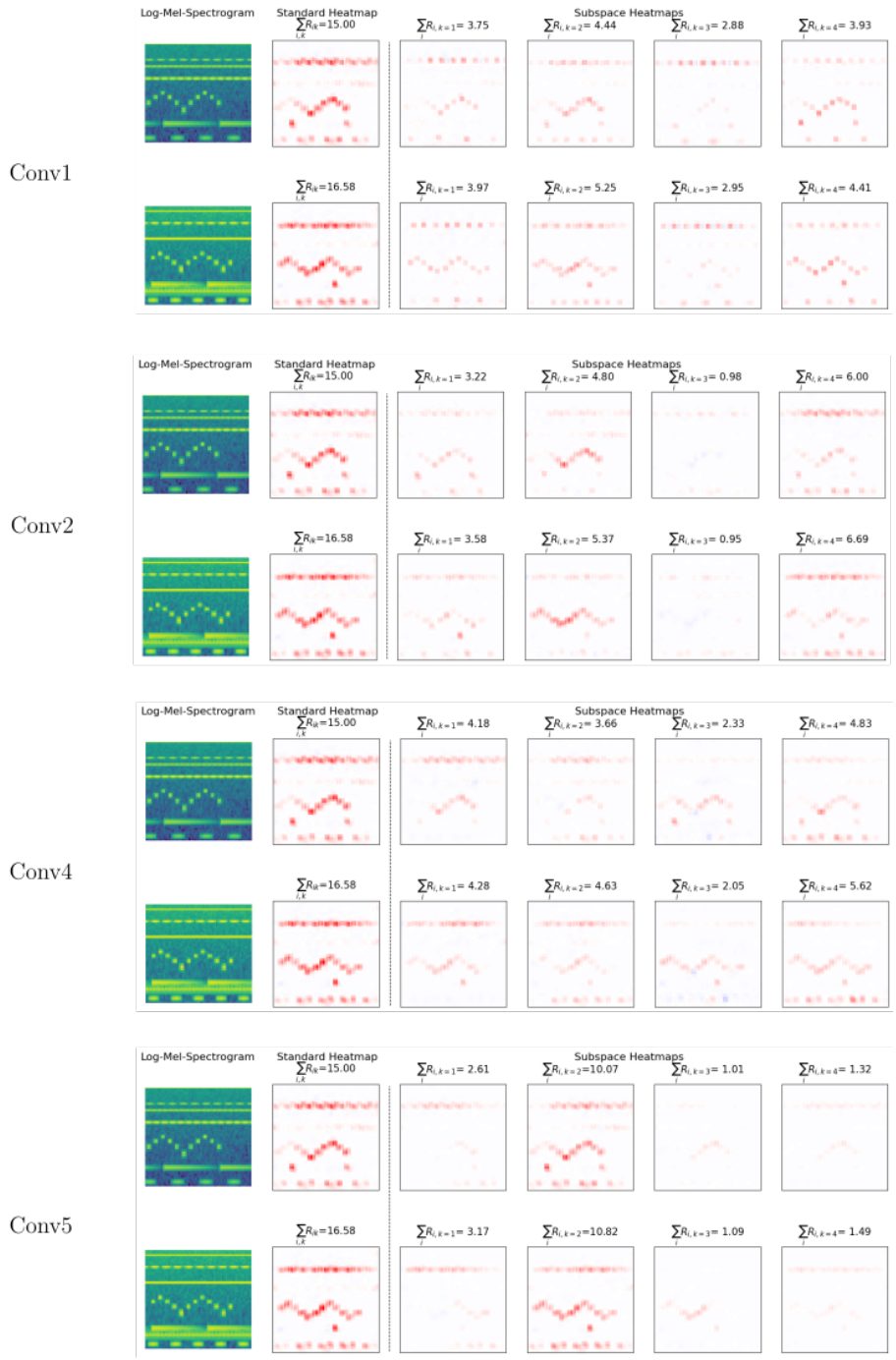



Figure C.3.: Multiple examples of the explanation disentanglement achieved with DRSA on toy class 2 at different convolutional layers. The order of images, and notation within their titles follows Fig. C.1.

## C.2. GTZAN Data

This chapter provides further examples of explanation components extracted on several genre classes. Starting off with Fig. C.4, the aforementioned 'chop' for genre class reggae is represented by subspace $k = 1$ and $k = 2$. Apart from the 'chop', through careful audible inspection, we can observe that subspace $k = 3$ represents the base line, and subape $k = 4$ the kick drums. Fig. C.5 displays concepts extracted for metal music. These components represent: low frequency base lines and kick drums ($k = 1$), vocals and e-guitar ($k = 2$), high frequent snare drums ($k = 3$), and e-guitar noise combined with kick drums ($k = 4$). Disentangled explanations for classical music are displayed in Fig. C.6. To recover: classical music achieved the highest $\Delta$AUPC score (refer to Fig. 4.10). The components seem to correspond to different patters, yet, domain knowledge and further examination has to be conducted to interpret the musical objects represented by the heatmaps.

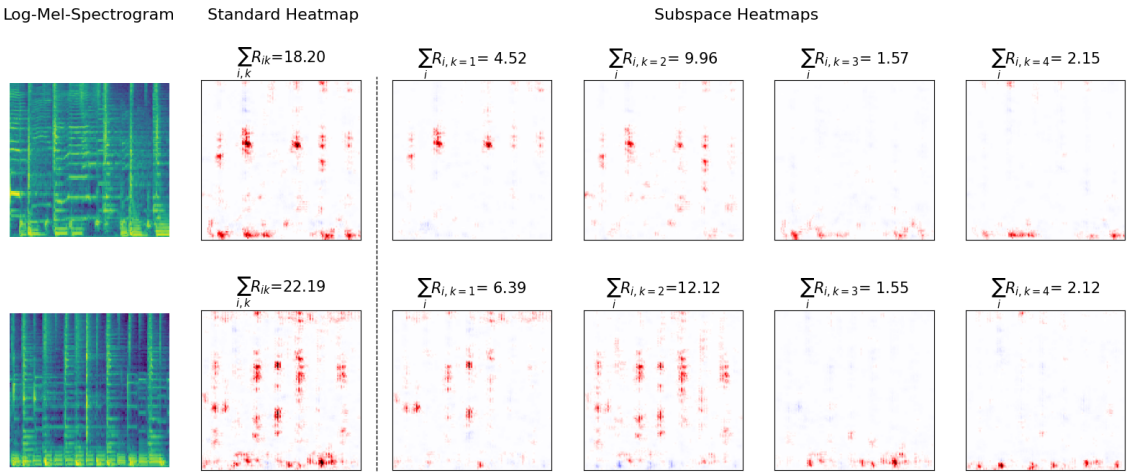



Figure C.4.: DRSA results on genre class reggae, with $K = 4$ subspaces at at Conv4. The order of images, and notation within their titles follows Fig. 4.6. 🔊: `gtzan/reggae`

Fig. C.7 and C.8 show explanations extracted for genre hiphop at Conv4, with $K = 2$, and $K = 8$ subspaces respectively. The samples are the same as used in Fig. 4.6. It is apparent that the component heatmaps are arguable worse than sub-explanations generated with $K = 4$ subspaces.

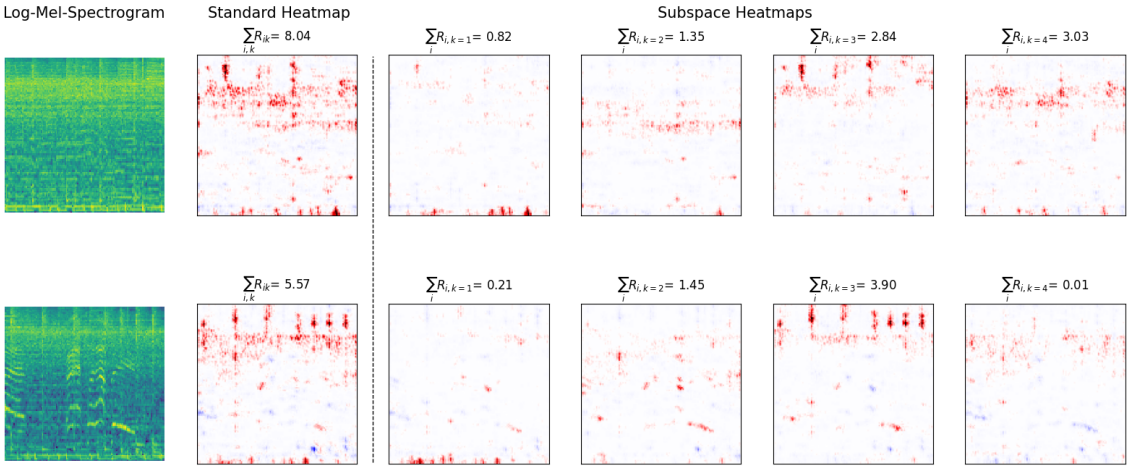



Figure C.5.: DRSA results on genre class metal, with $K = 4$ subspaces at Conv4. The order of images, and notation within their titles follows Fig. 4.6. 🔊: `gtzan/metal`

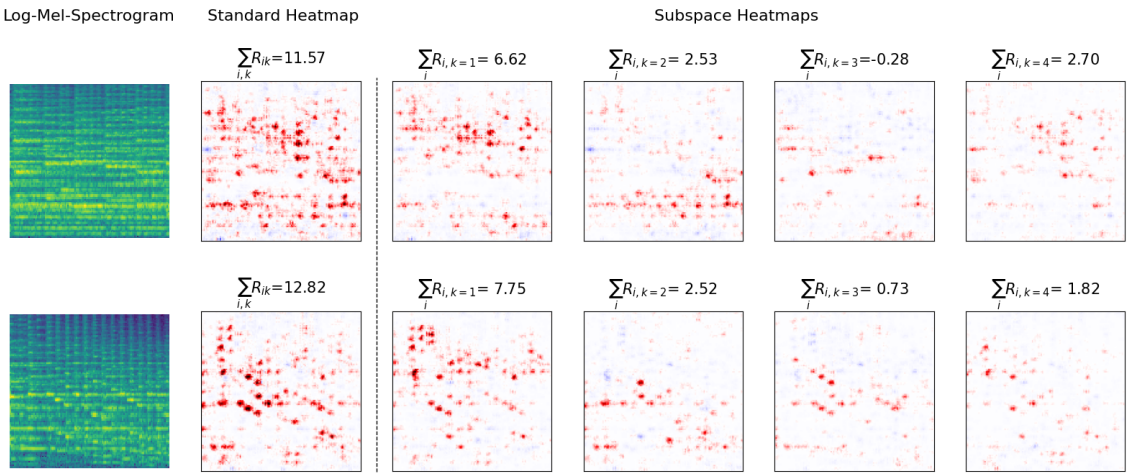



Figure C.6.: DRSA results on genre class classical, with $K = 4$ subspaces at Conv4. The order of images, and notation within their titles follows Fig. C.4. 🔊: `gtzan/classical`

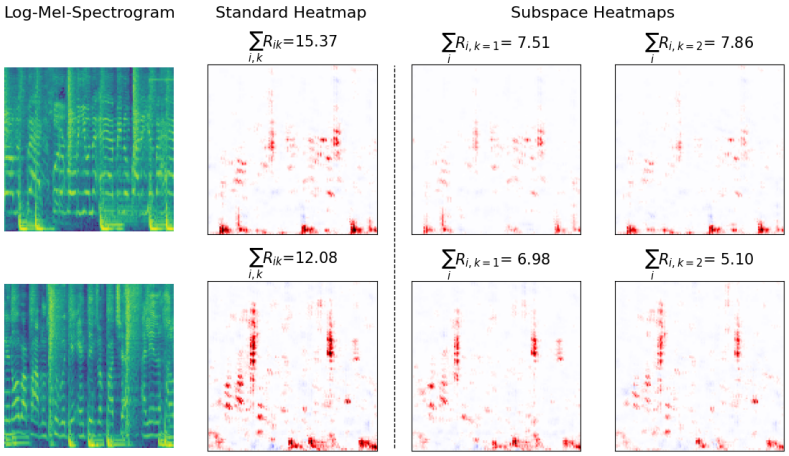



Figure C.7.: DRSA results on genre class hiphop, with $K = 2$ subspaces, extracted at Conv4. The notation within the titles follows Fig. 4.6.

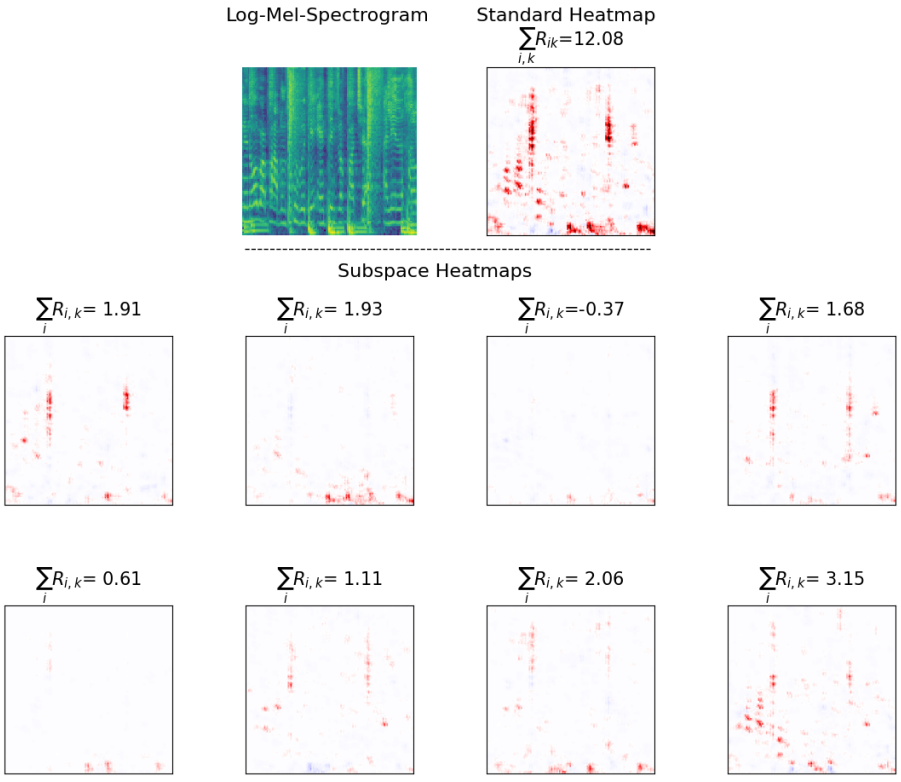



Figure C.8.: DRSA results on genre class hiphop, with $K = 8$ subspaces, extracted at Conv4. The notation within the titles follows Fig. 4.6.

# D | Synthetic Data Generation

This chapter elaborates on the generation process of the synthetic dataset. Information stated in this chapter is mainly extracted from [80]. As mentioned in Section 4.1.1, each sample is a combination of up to four class specific audio object, random sounds, and Gaussian noise with a noise strength of 0.1. Each audio object has a fundamental sound frequency $f$, which is sampled randomly form a predefined range. Refer to Table 4.1 for details. The fundamental sound signal $x[n]$, with $n \in N$, and $N \in \mathbb{N}$ being its total length, is represented by a periodic sine-wave with

$$x[n] = A_{mod}[n] \cdot \sin(\frac{2\pi \cdot k_f \cdot n}{N} + \phi_f)\,, \tag{D.1}$$

where $\phi_f$ defines the phase of the signal, and $A_{mod}[n]$ denotes the amplitude at time point $n$. The parameter $k_f$ defines the frequency of the signal according to

$$f = \frac{k_f f_s}{N}\,. \tag{D.2}$$

Since the sample rate $f_s$ is chosen to be 16000 Hz and the duration of each object is set to 1 second, Eq. D.2 simplifies to $f = k_f$. The modulating amplitude is characterized as a periodic sine-wave by

$$A_{mod}[n] = \max(0, \alpha \cdot \sin(\frac{2\pi \cdot k_a \cdot n}{N} + \phi_a) + \beta) \cdot \frac{1}{\beta + 1}\,, \tag{D.3}$$

where $k_a$ denotes the frequency of the signal. Furthermore, $\alpha$ is the maximum amplitude of the sine-wave that is sampled randomly within the range $[0.5, 1]$, and $\beta \in [0, 1]$ defines an amplitude shift which alters the modulation structure and is set differently between sound objects. For instance, $\beta = 0$ results in a half-wave rectified sine-wave. The fraction $\frac{1}{\beta+1}$ serves as normalization term, to assure all values $A_{mod}[n]$ in $N$ are within the range of $[0, 1]$. Furthermore, phases $\phi_f$, and $\phi_a$, are both sampled at random from the range $[0, 2\pi]$. This allows rhythmic structures to occur at different time points.

## D.1. Synthetic Toy Class 1

For the ease of understandability, Fig. D.1 shows one example per sound object of class 1. The fundamental sound of each signal is generated with Eq. D.1, by sampling its frequency from the predefined ranges stated in Table 4.1. The following sections elaborate on additional transformations, such as the generation process of modulating amplitudes.

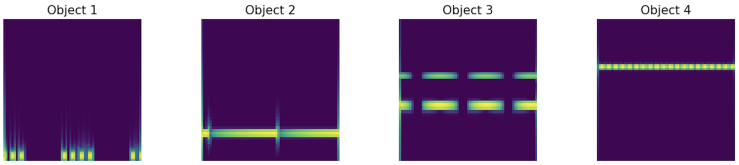



Figure D.1.: Class specific audio objects of toy class 1.

**Sound Object** 1

Amplitudes for this signal are generated by employing Eq. D.3, with $\beta$ set to 0. Thereafter, amplitudes are combined with the fundamental sound, and the resulting signal $x'_{11}[n]$ is masked according to

$$x_{11}[n] = x'_{11}[n] \cdot \mathbb{1}_{y[n]>0}\,, \tag{D.4}$$

where $\mathbb{1}_{\{\}}$ denotes an indicator function which is 1 if the statement in the subscript is true, and 0 else. The signal $y[n]$ is defined by Eq. D.3, with $k_a = 2$, and $\beta = 0$. In consequence, the operation performed in Eq. D.4, masks the rhythmic base signal at random locations by incorporating a random phase.

**Sound Object** 2

The modulation of this sound object is determined by a sawtooth function. The latter is defined as

$$As_{mod}[n] = \left| \gamma \cdot \left[ \left( \frac{2\pi \cdot k_s \cdot n}{N} + \phi_s \right) \mod 2\pi \right] \cdot \frac{1}{2\pi} \right|, \tag{D.5}$$

where $k_s$ denotes its frequency, $\phi_s$ is the phase, $\gamma \in \{-1, 1\}$ defines the direction of the sawtooth, and *mod* is the modulus operator. For this signal, $\gamma$ is set to 1, and $k_a$ to 2. The phase is sampled at random.

**Sound Object** 3

Initially, the base signal $x'_{13}[n]$, generated with Eq. D.1, is altered by adding the first subsequent harmonic to it. Harmonics are defined as integer multiples of the base frequency $f$ of some signal. Hence, it follows

$$x_{13}[n] = x'_{13}[n] + x_{h1}[n]\,, \tag{D.6}$$

where the frequency of the base signal $x'_{13}[n]$ is given by $f_{13}$, and the frequency of $x_{h1}[n]$ is

defined as $2f_{13}$. Magnitudes for the resulting harmonic signal $x_{13}[n]$ are provided by Eq. D.3, with a shift of $\beta = 0.75$.

**Sound Object** 4

This object defines a rather simple signal, where the amplitudes are given by Eq. D.3, with a shift of $\beta = 1$.

## D.2. Synthetic Toy Class 2

Exemplary audio objects of this class are displayed in Fig. D.2.

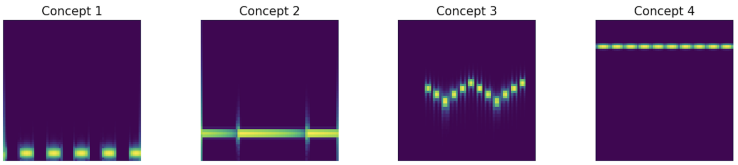



Figure D.2.: Class specific audio objects of toy class 2.

**Sound Object** 1

Magnitudes for this signal are represented by a half-wave rectified sine-wave, i.e., with shift $\beta$ set to 0 in Eq. D.3.

**Sound Object** 2

The amplitudes for this sound object are created according to Eq. D.5, with $\gamma$ set to $-1$, to obtain a decreasing sawtooth.

**Sound Object** 3

Sound object 3 alternates between a base frequency $f_{23}$, and an upper frequency $f_{max}$, which is set to $f_{23} + 600\,\text{Hz}$. To construct this signal, 4 separate sine-waves are created according to Eq. D.1, with evenly spaces frequencies between $f_{23}$ and $f_{max}$. The amplitudes are generated by a half-wave rectified sine-wave. Via controlled masking operations, each sine-wave is masked to eventually generate a modulating, and alternating signal, as depicted in Fig. D.2. This signal is restricted to include 12 modulating sounds. Through randomizing phase of the masking operations, various alternating shapes are generated.

**Sound Object** 4

Similar to sound object 1 of class 1, amplitudes are defined with a shift of $\beta = 1$.

### D.2.1. Final Sample Construction

Suppose some sample $x[n]$ corresponding to some class $c$, with $c \in C$ and $C = 2$. After superposing up to 4 sound objects associated with class $c$, $3 - 5$ random sound are added. The base frequency of each random sound signal is sampled from an exponential distribution that is governed by the probability-density function

$$f(f;\beta) = \frac{1}{\beta}\exp(-\frac{f}{\beta})\,. \tag{D.7}$$

Eq. D.7 models the probability density at a specific frequency $f$. The scale parameter is denoted by $\beta$ and set to 2000. Sampling frequencies from this distribution ensures a focus on lower frequencies, which was chosen due to the quasi-logarithmic mel scale. In consequence, random signals spread nicely across a mel-spectrogram after time-frequency transformation. Frequencies outside the range of $[1, 8000]$ are clamped. Eventually, a fundamental sound signal is randomly combined with a modulating amplitude with shift 2, and modulation frequency within the range of $[40, 100]$. In the end, Gaussian noise, with noise strength 0.01, is added to the resulting sample, and the signal is normalized by peak as defined by Eq. A.1.