POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Explaining black-box models in the context of audio classification

Supervisors

Prof. Tania CERQUITELLI

Phd Francesco VENTURA

Candidate

Giuseppe DE LUCA

Academic Year 2020/2021

Acknowledgements

My first thanks go to Prof. Tania Cerquitelli and Dr. Francesco Ventura who, on the one hand, gave me the opportunity to work on this project in a field that aroused all my passion and interest and, on the other hand, always knew how to accompany me and help me resolve my doubts and difficulties in a precise, clear and timely manner.

Thanks to my father Antonio, an example of total self-sacrifice at work and constant commitment, to my mother Franca, a loving guide and example of continuous attention, to my brother Gabriele, who physically accompanied me during my university years and who has always been an ambassador of family affection far from home. Their love and attention was never lacking a single day in the moments when we were far apart, and they were fundamental in my journey and its conclusion.

A thank you will never be enough for Chiara, constant point of reference and unconditional love. The whole journey would have been impossible without her. Her support has been fundamental throughout my university career and everyday life and not a single one goes by when I don't feel lucky to have someone like her by my side. Thank you for your love and patience. I am grateful for everything.

A kiss to the sky to my guardian angel who left too early to see me reach this goal. I am sure that wherever you are now, you are proud of me. I miss you so much.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the years, Artificial Intelligence(AI) has increased its importance and impact in today's society and we have also grown accustomed to accepting his decisions. Several aspects of our everyday life are based on AI decisions : product recommendations, friend suggestions, targeted advertising. In some important fields AI decisions have a life-changing importance, such as disease diagnosis or autonomous vehicle decision. In this fields failure is not acceptable. AI algorithms are very powerful in terms of prediction results , especially Machine Learning (ML) algorithms. The success of deep learning (DL) and neural networks (NNs) has lead to more and more research and industrial applications. However those algorithms suffers from opacity, that it we have difficulties to understand the reasons behinds , in some cases , crucial decisions. We have the risk of creating decision models that learn from data that contains human biases and prejudices leading to wrong and unfair decisions and moreover we can't resolve these errors because we don't really understand decision systems. This impacts not only on ethics but also on safety and industrial products. Explanations technology can help to create safer products and the results in scientific research, based on ML models, requires an explanations for trust and acceptance but also for help the progress of research. The main focus is that entrust decision to a system that is a black-box and cannot explain itself is a risk. Explainable Artificial Intelligence (XAI) propose to move to a more transparent and understandable AI. The goal is to build a suite of technologies that produce more explainable models without lower performances.

Figure 1.1
Google research with Explainable Artificial Intelligence terms

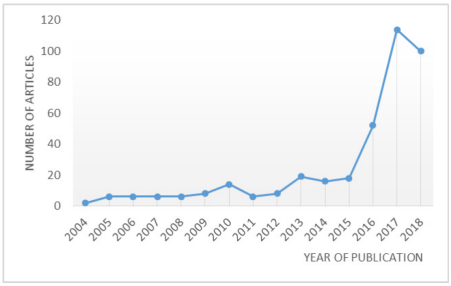


Figure 1.2: Articles about XAI by year(2004-2018)

In the last years XAI has become very popular in ML research community and also popular DL libraries have included their XAI libraries as we can see in Figure 1.1 that show the incremental trend in Google research including Explainable Artificial Intelligence terms and in Figure 1.2 the plot describe the number of articles written about XAI until 2018. This has led to the development of frameworks dedicated to producing explanations for black-box model especially in the context of image and text classification showed in chapter 2
What is missing in the panorama of Explainable Artificial Intelligence is a technique to produce explanations in the context of *audio classification*.
The main contribution of our work is to build and test for the first time an Explainable Intelligence technique that can produce usefull explanations of input audio clips to makes models of audio classification more understandable by humans.
Audios have a lot of representation each of which try to represent important features of the audio that can be used as input of a neural network in order to compute classifications as explained in chapter 3. Among all the possible representation we choose to feed as input to a classifier the Log-Mel Spectrogram of the input audio. *Log-Mel spectrogram* is a visual representation of the audio power as it varies over time at different frequencies and is deeply described in chapter 3.

The first phase of the work consisted of building and training an audio classifier. The neural network is builder upon a slightly modified version of VGG, popular convolutional neural network that showed high performances in computer vision task. The classifier takes in input patches of the input Log-Mel Spectrogram. The model architecture and training phase is described in chapter 3.

Since the input of the classifier is a visual representation of an input audio, the main work on this project consisted in the adaptation of a general explainable artificial intelligence frameworks in the context of *audio classification*. We called our proposed technique *A-EBAnO*.

A-EBAnO produces *local explanations* of the model by applying iterative perturbation on a set of *interpretable features* and computing the impact that the perturbed features have on the classification. The main characteristics of A-EBAnO are :

- Exploitation of inner knowledge of the model under analysis.
- Unsupervised extraction of interpretable features easily understandable.
- Visual and numerical explanation based on influence and precision of the input features.

The workflow of A-EBAnO is composed by several phases, and for each phase precise solutions have been found and adopted in order to have an novel explanations framework suitable for the explanation of audio classifications. A deep description of A-EBAnO is present in chapter 4. Many experiments have been performed on A-EBAnO, the results of which are reported in chapter 5 with some explanations examples and global statistics. These experiments emphasize the capabilities of A-EBAno to :

- Find features that have a positive, neutral or negative impact on the classification.
- Provide the most suitable explanation if we want to focus on time or frequencies as well as on the general spectrogram.
- Show if the model have cognitive bias due to the training phase.
- Compare models in order to understand what are the more trustable.

The final chapter of this paper present the conclusions on this thesis and some proposed improvement of the current implementation.

Chapter 2

Explainable Artificial Intelligence Overview

This chapter shows the needs for XAI methodologies, the different algorithms and techniques with which an explainable model can be characterized and a discussion over state of the art technique present in XAI literature.

2.0.1 Needs for eXplainable Artificial Intelligence

To provide an explanation we need an *interpretable* model. In machine learning interpretability means being able to produce an explanation in a way that is *comprehensible* to a human. Several reasons underlie the need for XAI methodologies.

Justify results First of all AI must provide justification to comply with the law, for instance with regulations included in General Data Protection Regulation (GDPR), that started to take effect in EU in 2018. When we think about an explanation for a decision, we generally want a justification for a particular result instead of understanding the inner workings of the process. XAI technologies provides information to explain results and a way to defend decisions, especially when these are strange or unexpected.

Systems control Things can go wrong at any time so a deeper knowledge of the behavior of the systems help to prevents errors or to correct them quickly in low criticality situations by uncovering hidden vulnerability. This allows advanced control of the systems.

Improve models Once that a model has been understood and explained the successive job of improvement is clearly facilitated and moreover we are able to

make it more intelligent. Therefore XAI can be the starting point for continuous iterations between human users and machine.

Learn new facts AI has been shown to have the ability to outperform humans in some specific areas. It would be desirable that the machine have the ability also to explain its knowledge in an interpretable way for us humans. Only explainable systems can do this.

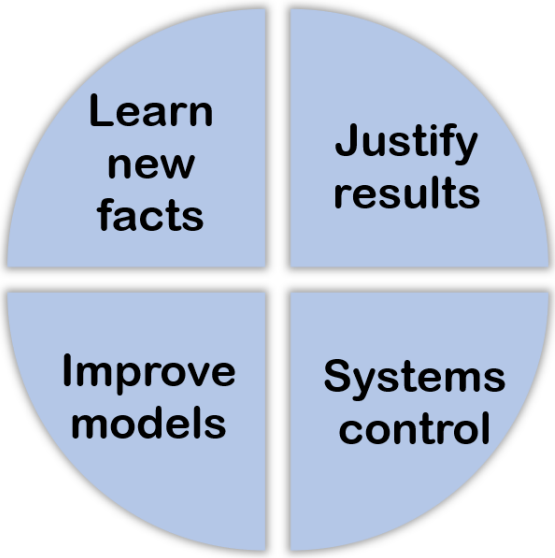


Figure 2.1: Reasons for XAI

2.0.2 XAI models and technologies

After discussing how XAI is useful to justify results, control systems and improves our knowledge, let's see how these goals can be achieved in terms of algorithms and technologies used, summarizing works present in [1] and [2].

At the state of the art we have to disposition models *intrinsically interpretable* for humans: decision trees, rulers and linear models.
Systems based on a *decision tree* produces decisional graphs in a tree form. Internal nodes represent test on the attributes and each branch exiting the node represent a different result. Every leaf represent a class label. Following the path from the root to a leave we are able to build the classification rules. They are widely used for their graphical representation.
Decision rules are functions that links together an observation to an action. The most common are if-then rules where in the if clause we have a conjunctions of

input variables and the then clause represent a label. Their strength is their textual representation.
Linear models are capable to show how influential is an attribute for the final decision. If an attribute increase the model's output than it have a positive influence and vice versa. We are able also to understand how much is the influence of the input attributes when we have unexpected changes between an expected output and the result of the model.
Exploiting the inner capability of this methods to be easily understandable we can build a *transparent box design* for the model to be understandable on its own. The biggest disadvantages of these methods is that they lack accuracy. Therefore, to improve accuracy, more complex methods are used and more is the complexity, less is the understandability of these methods. Examples for this types of models are Neural Networks, Deep Neural Network, Tree Ensembles , Support Vector Machines.
An approach in XAI is to don't change these high complex black-box methods but to use a reverse engineering process to create post-hoc explanations. These type of methods are the most common in XAI techniques.

Our objective could be to understand a model globally or only one specific result. *Global explanations* shows the whole logic of a model and all entire reasoning leading to all possible different results. A global interpretability is very hard to achieve especially for methods that have in input a huge number of parameters. We can refer to these methods as *black-box model explanations*.
Following the human habit of focusing on a specific part of the model to understand the whole of it, a local interpretability may be easier to implement.
Local explanations is focused to explain and justify the result for a single specific instance . For example to explain the decision of an image classifier, a common approach is to alter parts of the image to see if those pixels have any influence on the final label classification. For every image parts that are altered differs from the parts of the other images, so the explanation is only locally important for that image. We can refer to these methods as *black-box outcome explanations*.
Local explanations are the most used methods to make Deep Neural Networks more understandable, but usually this type of approach are applicable to any type of models, guiding us towards a further classification of the XAI technologies.

Model specific methods are linked to a specific model. The drawback of these methods is that we have limited choices when we build the explainable methods.
Model agnostic methods separate predictions from explanation and usually use post-hoc interpretations. They are the most common developed methods in recent times.

2.0.3 State of the art XAI techniques and works

In this section are firstly presented state of the art technique used in eXplainable Artificial Intelligence and some frameworks with characteristics related to our technique.

The variety of techniques used to build explanations is large.

Visualization The first natural idea to understand a model, especially Deep Neural Networks, is to visualize a representation of his inner units. This type of techniques are applied to supervised learning models.
Surrogate models are inherently explainable models, like Decision Trees and Linear models, that are trained on the result of a more complex black-box model, in order to understand and explain the latter, at the risk that the simpler model is not enough representative of the more complex model.
Partial Dependency Plot (PDP) is a graphical representation of partial correlation between input variables and a possible outcome.
Individual Conditional Expectation (ICE) extend the raw representation of PDP, revealing links and difference between every single input variable. Another popular technique is *Salient Mask* (SM),used especially with images and texts. This technique use a mask to highlight important feature of the input extracted from the model.

Extract Knowledge During training phase a model acquire new and improved knowledge at every step, modifying the representation of his hidden layers. The effort here is to represent in an understandable form this hidden knowledge, decoding his hidden representation. We talked before about Rule Extraction.
Another technique is to compress and transfer learning from a deep network to a shallow network.

Feature Influence This type of models modify the internal representation of a model or the input in order to understand how much a change in a feature influence the model performance. Often this type of models are also visualized.
Sensitive Analysis (SA) show how good is the model after introducing perturbation on the input or hidden layers. It is considered an agnostic explanation technique.
Variable Importance show the importance of a feature for the final prediction of a complex model. Modifying the value of important features increase the error on the prediction so we can discriminate from important, neutral or not important features.

Examples Explanations based on examples don't make some modification to the model or its inner representation but instead they highlight some input data from a data-set and use them to explain the model behaviour.
Prototypes are representative data for the classification and more an input is similar to a prototype higher is the probability that it is assigned to the same class. To avoid over-generalization is important also to show some *exceptions*: instances that are not represented by the chosen prototypes.
Another goal is to explain the minimum change in the input that lead to a different result than leaving the input unchanged. This type of technique is called *Counterfactual Explanations*. Models that utilizes visual or textual representation for explanations can be called *Black-Box Inspection models*.

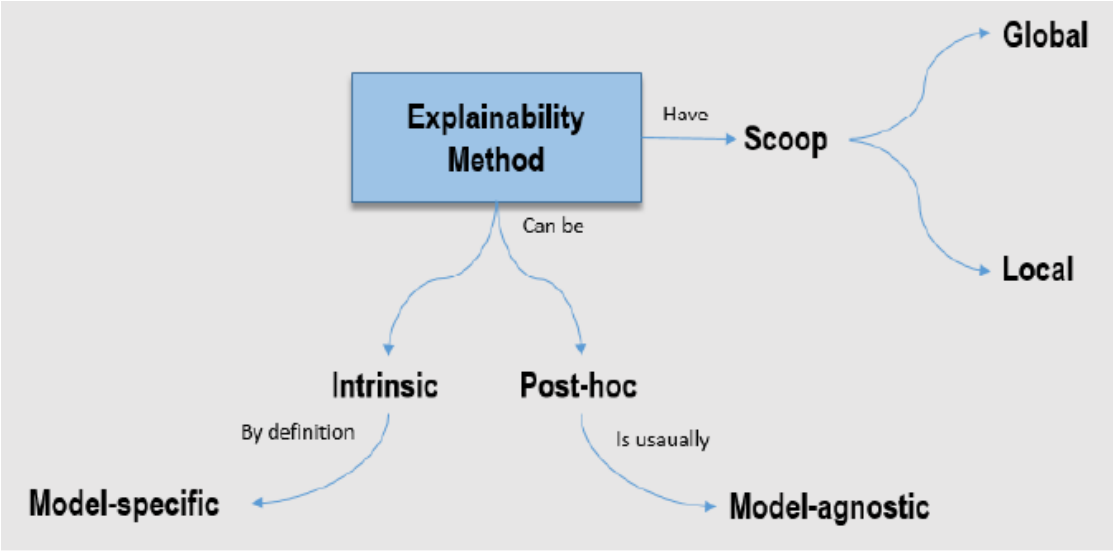


Figure 2.2: XAI methods taxonomy

Finally, it is important to focus on the type of input data for classification models.
Every type of a data has a different interpretative capacity for humans. Most of the models in XAI works on very interpretative input as *image*, *text* and *tables* that are easily understandable.
Our effort in this project is to work with a type of data that is not yet been explored in literature, *audio data*.

Since in literature are not present works that have as topic the explanation of *audio classification*, we present works that are the most related or otherwise have

similar features to the model we developed. The frameworks described focus on the explanations of *Deep Neural Networks* results or *agnostic* methods that use *visualization* to produce *local post-hoc* explanations for image or text input data classification task.

Visual attention for Image Caption Generation In [3] authors describe their attention based model to automatically create a caption for an image. Their model is build upon a Convolutional Neural Network(CNN), that takes in input an image and build a set of feature vectors extracting features from low convolutional layer, focusing in more specific parts of the input image. Than features vectors are decoded into captions trough a Long Short Term Memory (LSTM) Network, that produces a word for every time step.
Authors wanted to add interpretability for their model so they wanted to visualize the attention learned by the model. In order to reach their goal, they upsampled attention weights and they added also a Gaussian filter. In this way it is possible to understand especially why errors are made by the model. Figure 2.3 show an example of correct attention explanation while Figure 2.4 report the explanation of a wrong attention. Figures are taken from paper [3].

Figure 2.3: Correct attention explanation.

Figure 2.4: Wrong attention explanation.

Rationales for text predictions In [4] is presented a way to incorporate justifications for results of a model directly in the learning problem. Authors call their justifications rationales. The phrases of the text to be rationales must be short and coherent and if used as input for the classification task they must sufficient to reach the same result of the original text. In order to understand how good it's their model authors highlights rationales for sentiment of different aspect with different colors. In Figure 2.5 taken from [4] are presented some examples of rationales.

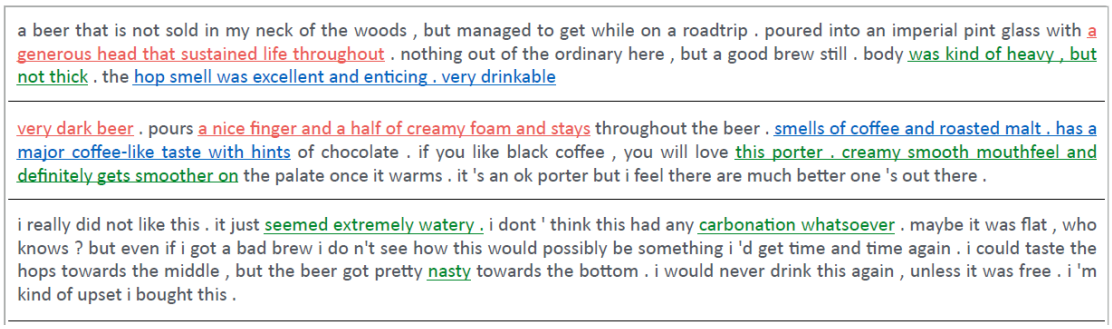


Figure 2.5: Rationales for sentiment of various aspects : appearance is red, smell is blue and palate is green.

Class Activation Mapping (CAM) In [5] authors shows a generic algorithm called Class Activation Mapping for Convolutional Neural Network. They exploits the fact that convolutional units can detect object in an input image and using Global Average pooling layer the ability to localize is kept intact up to top layers without being lost through Fully Connected Layers. CAM for a specific label represent the active parts of the image that are responsible for a particular outcome and its produced by doing a linear combination of the final features maps with the weights learned in the last layer. Finally to be visualized it is normalized between 0 and 1.

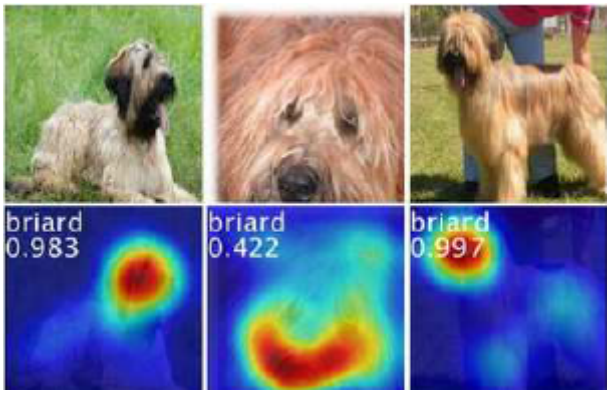


Figure 2.6: CAM for label "Briard". Taken from [5].

Gradient-weighted CAM (Grad-CAM) Authors of [6] focus on information that comes from class-specif gradient in order to find important parts of the input image for the classification. They propose Grad-CAM as visual explanation for any type of CNN. In contrast to CAM [5], that need a retraining for the model after changing Fully-Connected Layers with pooling layers, this approach can be

used with any CNN without retraining. Authors highlights a second approach, Guided Grad-CAM, that contains high resolution and fine details in contrast to low resolution results of Grad-CAM.

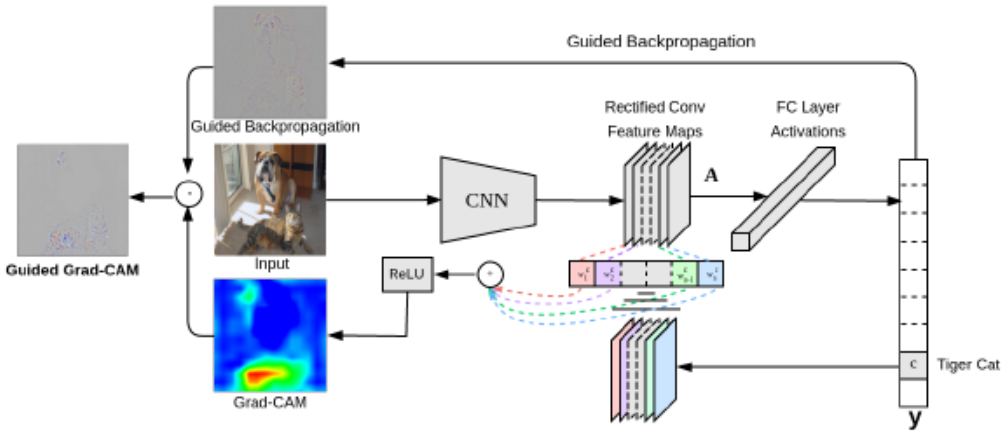


Figure 2.7: Grad-CAM Overview taken from [6].

Explanations by Perturbations In [7] authors propose a framework to build explanations as meta-predictors. An explanation is a rule that predicts the result of a black-box classifier with certain input. One advantage it is that is possible to use explanation's ' prediction accuracy as a way to measure its faithfulness. Their saliency maps can show the reason behind a classification without highlights evidence that are not essential. They discuss also various effect on classification of different types of perturbation explaining where a classifier look by finding which part of the image change method's score after being perturbed.

Figure 2.8: Example of learned mask. From [7].

Figure 2.9: Effect of perturbation on classification score. Taken from [7].

Local Interpretable Model-Agnostic Explanations (LIME) Authors of paper [8] propose LIME as a way to explain outcomes of any type of classifier, thus it is model-agnostic, building a model that is locally interpretable around the prediction. LIME can be applied both on images and text data. They consider a binary vector of words indicating if a word is present or not, an interpretable representation of a text. Instead for a image, the representation is a binary vector indicating the presence of group of similar pixel near each other (super-pixel). In order to produce an explanation, small parts of the input data are perturbed and associated with a label. They made predictions interpretable setting a max number of word or patches of pixels. Pixels that are important for prediction are showed and pixels not important for the prediction are greyed out.

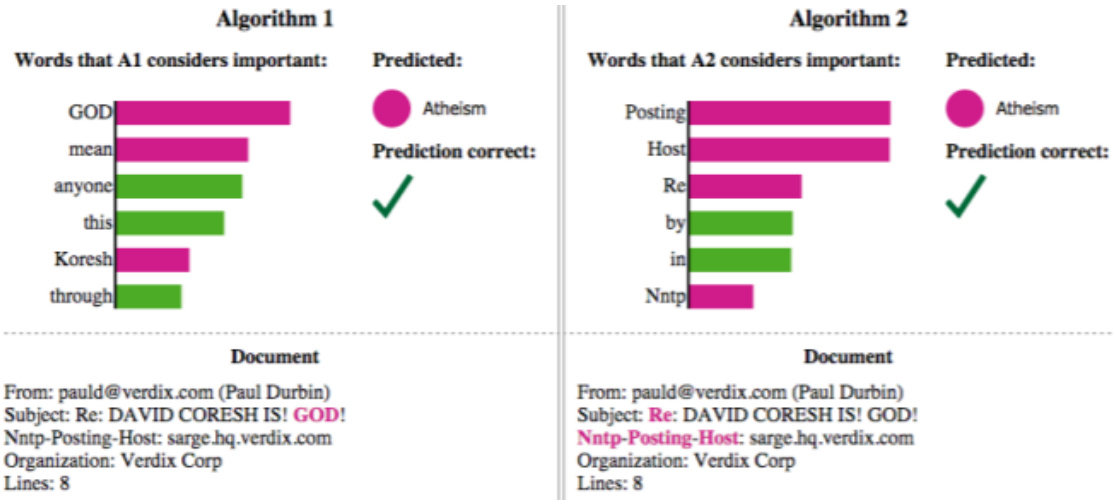


Figure 2.10: Individual explanations for two different models trying to understand if a text is about "Christianity" or "Atheism". Bar chart indicates word importance and color represent the most representative class for that word. From [8].

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 2.11: Examples for 3 different classification explanations on a single original image. From [8].

Chapter 3

Audio Classification and case of study

In this chapter are presented a general overview on Audio classification followed by a precise description of the audio representation used as input in our proposed XAI technique and finally we present the neural network and training phase of the network used as black-box model to produce the explanations.

3.1 Audio Classification

Audio classification is an application of machine learning and pattern recognition where an *audio signal* is associated to corresponding acoustic events in an auditory scene by use of a class label [9].
It is utilized in various applications such as automatic tagging of audio files for audio retrieval, home-monitoring environment for assist elderly people living alone or for smart-home, it finds applications in surveillance and recognition of animal and bird species [10].

Audio signals presents a wide range of groups and sub-groups of most of which intersect [11]. Approximately *audible sounds* can be divided into 3 groups :

Speech Sound produced by an human to communicate something.

Music Human-made sound with some instruments, including also human body , to express feelings or emotions of some nature.

Environmental sound It can be subdivided into different categories such as noise, natural sounds and artificial sounds.
Noise can be defined as as spontaneous or pseudo random signal with a precise distribution of energy, that can be uniform, like happens in White noise, or non-uniform. Noise can be also defined as unwanted or unnatural sound from a perceptive point-of-view.
Natural sounds are non-human produced sound that belong from nature or environment of nature. Examples are sounds created by animals, air, climate.
Artificial sounds are the opposite of natural sounds and they are produced or influenced by human. Examples are sounds produced by cars, houses, machines and technology [11].

These different types of sounds have led to the development of different fields of research. The most relevant are :

Segmentation Preprocessing step in audio analysis whose aim is to find portions of the sound with homogeneous characteristics in order to divide parts that represent different types of sounds like speech, music and silence.

Speech recognition Syntactical recognition of words produced during speech and includes language and speaker's sex classification and emotion retrieval.

Music information retrieval Deals with the classification of pieces of music or different instruments , artists or music genres.

Environmental sounds recognition The most diverse research field because it includes the analysis and recognition of all types of sounds different from speech and music [12]. Audio present several aspects that can be captured. A large variety of *features* have been designed in order to capture all these different aspects and they belong to very different *domains*.

Audio analysis features are:

- *Zero Crossing Rate (ZCR)*:
  it is defined as the number of zero crossing on time domain within one second. It is very cheap and simple to compute.
- *Amplitude Descriptor (AD)*:
  the audio signal is divided into segments with low and high amplitude using an adaptable threshold. It is mostly used in the classification of animal sounds.
- *Short-Time energy (STE)*:
  it describes the envelope of a signal as the mean energy per frame.

- *Linear Predictive Coding (LPC)*:
  it is utilized in speech recognition and it exploits the source-filter model for speech.

- *Subband energy ratio*:
  it gives approximation of the audio signal energy distribution of the spectrum.

- *Spectral Flux (SF)*:
  it describe changes in the shape of the audio spectrum when it vary over time. A signal with low SF is nearly constant while high SF means abrupt changes in the spectrum.

These features belongs to *temporal domain* and *frequency domain*. They are extracted for raw audio data ,representing physical properties of the signal and they are based on different analysis techniques.
LPC exploit *autoregression analysis* where the value of each sample of the signal is estimated by a linear combination of shifted values of the same signal while SF is based on *Short-Time Fourier Transform* for the computation of the spectrogram.

In frequency domain there are other features that exploit *psychoacustic* properties of the sound having semantic meaning in the context of human auditory perception [12].
Examples of these features are *brightness*, correlated to the distribution of high and low frequencies on the signal; *tonality*, used to distinguish noise from tonal sounds; *loudness*, that simulate the human sensation that order sounds from soft to loud; *pitch*, that imitate the human capability to distinguish a low sound from a high sound.

Another important domain of audio features is *Cepstral domain*.
Cepstral features represent log magnitude spectrum of the sound with smooth frequencies and they are obtained computing the Fourier Transform of the logarithmic magnitude of the spectrum of the original signal.

Mel-frequency cepstral coefficients (MFCCs) One of the standard techniques in audio retrieval. MFCCs are obtained by converting Fourier coefficients to Mel-Scale, logarithmicizing and compressing them to eliminate unnecessary information. The first few compression components are usually used to represent the shape of audio spectrum. Several variations of MFCCs have been studied in order to be applied in scales different from Mel-Scale but cepstral coefficients based on Mel-Scale are the most popular used [12].

Audio representations in time and frequency domains provides different information about the signal from a physical point-of-view. Time information produce measurable data about the signal and frequency methods give insight on the nature of the physical phenomenon producing the sound, analyzing the power of the different constituent frequencies of the signal.
However analyzing input audio using tools that exploit only one domain implies assuming that the signal is produced by a *stationary* phenomenon. Real life audio signals very often have characteristics that vary over time.
This leads to the need for a different class of techniques when dealing with *non-stationary* signals [10].

Wavelet based techniques Wavelet transform is a mathematical method utilized in the decomposition of a signal into various components with finite energy and it offer representation both in time and frequency. An analytic function admissible as 'mother wavelet' [10] is used to produce daughter wavelets by scaling and shifting in order to better focusing the signal in time-frequency space. Wavelet based strategies are used in music classification or to de-noise audio signals [11]. One of the strong points of wavelets is that they have a very good capability in recognizing impulsive sounds like gunshots. In general performances of wavelet features is comparable with MFCCs performances.

Sparse representation techniques They are based on a sparse representation of the audio signal obtained with a greedy algorithm. The representation is based on atoms in an overcomplete fixed dictionary [10]. Features based on this algorithm have slightly better performance respect to MFCCs. In order to improve performance it was proposed to use a dictionary signal dependent or to use a large number of atoms in the signal decomposition. This types of features are able to produce information on high time-frequency resolution and they improve audio classification performance when used in combination with MFCCs.

Power spectrum techniques Spectrogram deliver very important informations about energy of the audio signal in a specific frequency and time region. A lot of studies was done in order to better understand the influence of all the parameters involved in the calculation of an audio spectrogram like signal length, sampling rate, type of windowing function. One of the strengths of spectrogram is that it is a *visual* representation of the time-frequency distribution of an audio signal. This has been exploited in order to find and develop visual features extracted from the spectrogram that can be used in combination with Artificial Intelligence approaches to build audio classifiers. Spectrogram features are used in music instrument environmental audio classification where these features perform consistently better against MFCCs [10].

Machine learning methods used in audio classification are :

K-nearest neighbor (KNN) KNN algorithm assumes that similar audio data are near each other. For each training data the distance to all other training data is computed, listed , sorted and the first K points are selected. K is chosen after experimental study. Test data is labeled with the most represented class of the K chosen point.

Naïve Bayes This classifier is based on the Bayesian Theorem. It requires estimated prior and conditional probabilities of the input data in order to compute probability of input to belong to a certain class, called posterior probability. It is also based on the strong assumption that the conditional probabilities of the audio variables are statistically independent. This technique is very effective when input dimensionality is large.

Support Vector Machine (SVM) SVM approach classification task in a very direct way: the input data are mapped to an higher dimension and it search for a hyperplane that separates classes in this enlarged feature space. If its possible to find a separating hyperplane then it describes an infinite number of hyperplanes that can be used to separate classes and choose among them the classifier with ideal boundary margins. Boundaries between classes can be either linear or non-linear based on the properties of the input data.

Decision Tree (DT) Training data is used to build the DT than in order to classify a test data we have to follow a path starting from the root node of the decision tree. Every non-leaf node we encounter represent a function with different branch for each value it can receive. We continue on the branch that represent the characteristic values of the test data arriving at a leaf node representing the class label of the input.

Convolutional Neural Networks (CNNs) CNNs are composed by a large number of working units called neurons, aggregated in different types of layers with specific task. Convolutional layers aim at learning features from the input and pooling layer down-sample input along time and frequency dimension respectively [13]. In most of CNN architectures the final classification is linked only to the last convolutional layer. Alternatives approach process activations also from all intermediate features map, or split two dimensional convolutions into two one-dimension convolution in order to better separate transient and long-term characteristic of the input sound. Modification on traditional CNN are done in

order to learn filters that are frequency-aware or adding first and second order time derivative of spectrogram based features in order to detect impulsive sounds [13].

Feedforward Neural Networks FFNNs are one of the oldest machine learning models. They are composed of joined *Perceptrons* into layers. Perceptron is the nearest representation of a neuron. It is composed by multiple inputs, a summator to sum inputs multiplied by their *weights*, followed by an activation function before the output layer. All layers node are fully-connected and activation goes from input to output without back loops. Layers between input and output are called hidden layers. They are usually trained using back-propagation.

Recurrent Neural Networks (RNNs) RNNs are unidirectional networks in which every neuron in hidden layers receive input with a predefined time delay. They are used when we need a model that are able to access information from current and previous steps, like speech recognition tasks. This networks are capable to share parameters and weights of every step in time but RNN cannot remember informations very distant in time.

3.2 Log-Mel Spectrogram

In recent years, after having observed the enormous benefits brought by the advents of Deep Convolutional Neural Networks for image classification, several studies have tried to understand if this type of models could have equally satisfactory performance in the audio classification and audio event recognition in the sense of the ability to *identify a sound* from an audio data.

Various types of input for classifiers, as well as feature extraction techniques have been explored in several papers (section 3.1) and we found it particularly interesting to analyze as a technique the use of segments of a *Log-Mel Spectrogram* as input of the classifier. But what is and in which way a Log-Mel Spectrogram is

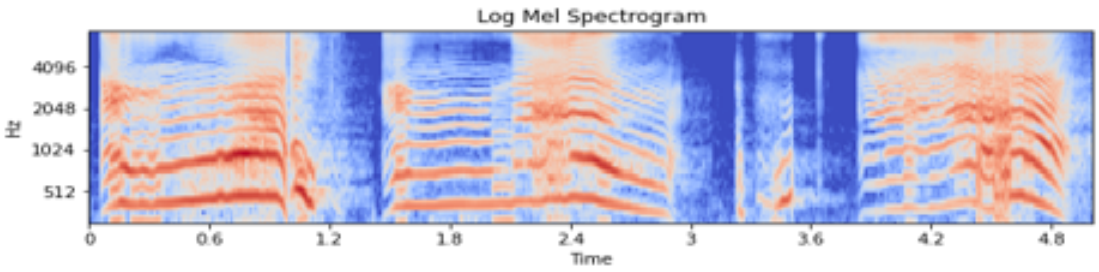


Figure 3.1: Log-Mel Spectrogram of Baby Crying

computed ?
An audio is a *signal* in which the quantity that change over time is air pressure . In order to digitally analyze this type of information the first step is *sampling*. We take samples of the air pressure in different instants of time, called *temporal sampling interval*. The reciprocal of temporal sampling interval is called *sampling rate*. Sampling rates used most frequently are 16000 samples per second (16 kHz) or 44100 samples per second (44.1 kHz).
After sampling the signal what we get is a *waveform* for the signal. After working in time domain we must move into frequency domain.
An audio signal is formed by a lot of sound waves with single frequency and it can be decomposed into a series of cosine and sine waves, which can be recomposed in the original audio signal. This is possible thanks to a very important mathematical formula called *Fourier transform*. We have available a very efficient algorithm for the calculation of the Fourier Transform called *Fast Fourier transform*. The result after the Fourier transform is called *spectrum*.
In order to represent the spectrum of the signal as it vary over time , the fast Fourier transform is calculated on windowed segments of the signal overlapping each other. This algorithm is called *short-time Fourier transform* and the result is finally the *spectrogram* of the signal.

As human beings we have a limited perception of frequencies and amplitudes so the y-axes of the spectrogram is converted to *logarithmic scale*. We are not able to perceive frequencies in a linear way. We can distinguish two frequencies very well if they are low while at high frequencies it is impossible for us. In 1937 Stevens, Volkmann, and Newmann proposed the *Mel Scale* that is a scale of the *pitch* of the sound(Figure 3.2). Equal distance on the pitch are perceived equally distant by the listener. Conversion between Hertz(f) and Mel(m) is defined as:

$$m = 2595 \log_{10}(1 + 700f)$$

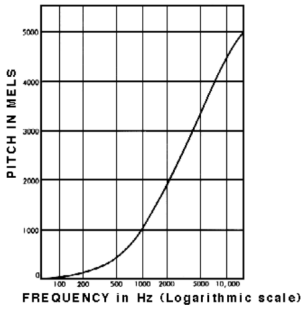


Figure 3.2: Mel Scale

Summarizing a Log-Mel spectrogram is a spectrogram with a logarithmic scale y-axis and where the frequencies are changed to the Mel scale.

3.3 Case of study: VGGish

In their work [14] authors, using Log-Mel spectrogram as input, show that state-of-the-art image neural networks achieve excellent results on audio classification. In particular, using VGG [15] they reached an average value across all classes of *AUC* equal to *0.911*. AUC is area under the Receiver Operating Characteristic (ROC) curve, that is , the correct accept rate as a function of false accept rate. Perfect classification achieve AUC of 1.0.

In [14] S. Hershey and other have fed as input on the most used Neural Networks non-overlapping frames of 960 ms each taken from an audio. Every frame is decomposed with a short-time Fourier transform using 25ms windows at time step of 10 ms. The spectrogram is than integrated into 64 Mel-spaced frequency bins covering the range 125-7550 Hz. The final input for all the networks is Log-Mel spectrogram patches with a shape of 96x64.Figure 3.3 is a representation of the process. An audio waveform is taken in input, Log-Mel spectrogram is computed and then divided in patches.

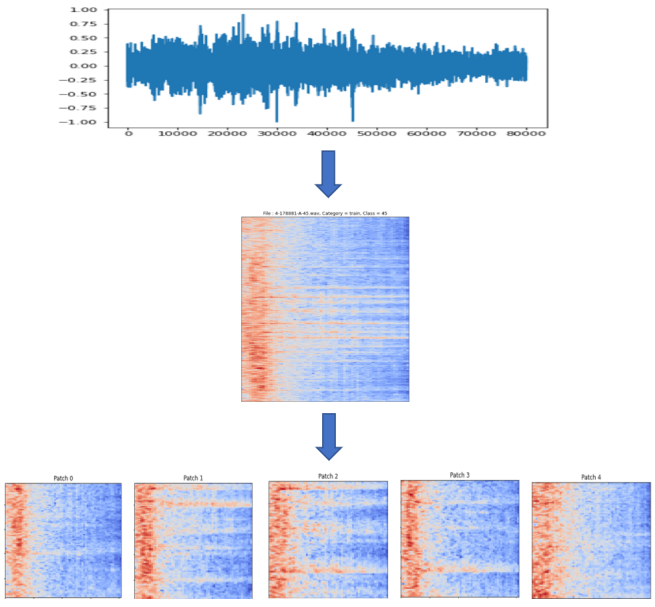


Figure 3.3: Model Input for 5s Audio

Among all the analyzed networks we decided to focus on a slightly modified version of VGG. We use a TensorFlow implementation of this model called *VGGish*. VGGish has the following changes compared to VGG (configuration A) [15]:

- Input size changed to 96x64 for Log-Mel spectrogram patches;
- Only four group of convolution/maxpool layer instead of five;
- 128-wide fully connected layer at the end. This acts as a compact embedding layer.

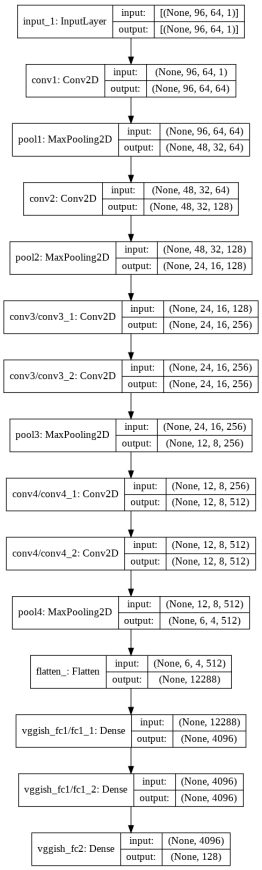


Figure 3.4: VGGish Architecture

Vggish can be used as a part of a large model. We treat Vggish as a warm start for lower layers of a model that takes audio features as input and add more layers on top of the Vggish embeddings because we want to train our model with Datasets very different from the one used for Vggish training.
Vggish model definition defines layers up to and including the 128-wide embedding

layer. In the context of neural networks *embeddings* are learned continuous vector representations of discrete variables and their strong point is that they are low-dimensional so they can reduce dimensionality of the input. Embeddings are learned on a supervised task and they form the weights of the network which are tuned with the aim to minimize loss on the task.
The embedding layer of Vggish does not include a non-linear activation, so before using embeddings as input of our model we have to send the embedding trough a non-linearity.
We choose the rectified linear activation function (Figure 3.5). If the input is positive it will be output directly, otherwise if the input is zero or negative it will output zero. An unit that use this activation function is called rectified linear activation unit ($ReLU$).
The output of ReLU than pass through 3 consecutive *500-wide Fully Connected Layers*. In this type of layers every neuron of a layer are connected to every neuron of another layer.
Output of the last fully-connected layer goes to a *Dropout* layer. This type of layer randomly set input to 0 with a chosen frequency. We set this frequency to *0.2*. Dropout layer helps to prevent *overfitting*, a dangerous condition in which the network is not able to generalize well on new data, different from data used for training, leading to a less accurate prediction. At the end the output of the classifier is calculated by a *Sigmoid* function (Figure 3.6). For every input the Sigmoid returns a value between 0 and 1 that represent the probability of the data to belong to a certain class. We decide to use this type of function because we desire to have a probability for every class separated from the probability of other classes as an audio can certainly contain more than one sound within it.
In Figure 3.7 we can see the final and complete architecture of the classifier. It will be used as the model for the explanations that we will create.

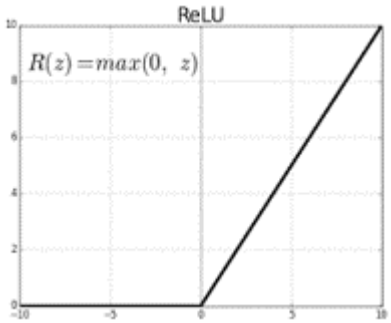


Figure 3.5: ReLU Activation Function

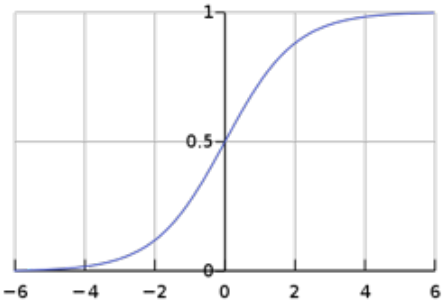


Figure 3.6: Sigmoid Activation Function

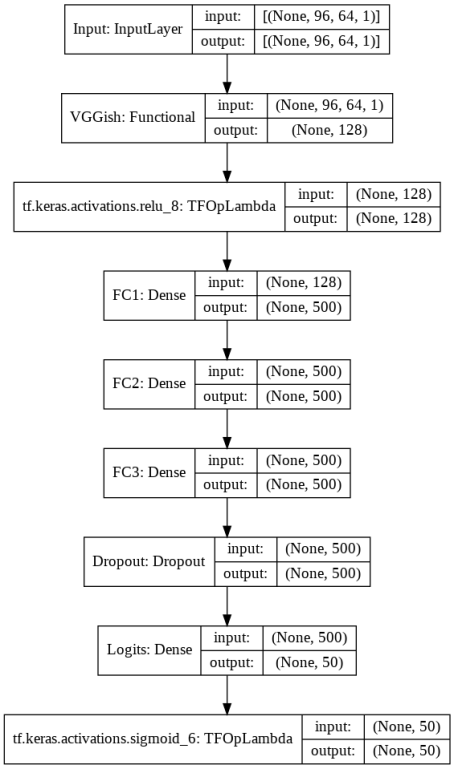


Figure 3.7: Classifier Architecture

3.4 Classifier Training

Machine Learning algorithms can be divided into two group according to the learning methodology.
Supervised learning is characterized by the use of a Dataset that contains data linked to their *label*. The Dataset is often divided into *training set*, that is feeded into the classifier in order act as a 'guide' to the model, that learn some characteristic from it and use the learned knowledge to produce classification on a *test set*, independent from the training test. The model is validated on a *validation set* with different possible strategy: subdividing training set into fixed training set and validation set or using *K-Fold Cross Validation*, in which the Dataset is divide in k group with the same size and for k times the model is trained in one of the groups and validated on the others.
Unsupervised learning is instead based on *unlabelled data* and the objective of the algorithm is to find group in which input data should be divided analyzing similarities and difference among them. In our work we use a *supervised learning algorithm* and we choose the followings as training Dataset:

1. Environmental Sound Classification 50 (ESC50)
2. UrbanSound8k

Both Dataset are composed by clips of *environmental sounds*. We choose them because we want to test our technique on sounds more related to every-day life rather than on more task specific types of sound like music or speech. After choosing the classification and learning algorithm more suitable for our task, the next phase is the *model fitting*. A classification model is a mathematical model that is characterized by important *parameters* that had to be tuned in order to obtain a good performance for the input classification. During this phase the model learn to classify the analyzed data.
During the training of our classifier we used the following parameters:

- *Learning Rate* : $1.5e^{-4}$
- *Optimizer* : Adam (epsilon = $1e^{-8}$)
- *Loss* : Categorical Cross Entropy Loss
- *Classification Metric* : Categorical Accuracy
- *Epochs* : 100

A fundamental choice is that of the metrics to be used to evaluate model classification. Each type of input and task has a dedicated set of metrics to use. On the test set we used on F1-Score as metric to valuate the performance of the classifier. *F1-Score* is the harmonic mean between precision and recall.

- *Precision* : $\frac{TruePositive}{TruePositive+FalsePositive}$
- *Recall*: $\frac{TruePositive}{TruePositive+FalseNegative}$
- *F1-Score*: $2 * \frac{Precision*Recall}{Precision+Recall}$

3.4.1 ESC50 Training

Environmental Sound Classification 50 [16] contains 2000 5s recordings that belongs to 50 different classes, with 40 clips per class.

The classes present are : Dog, Rooster, Pig, Cow, Frog, Cat, Hen, Insects, Sheep, Crow, Rain, Sea waves, Crackling fire, Crickets, Chirping birds, Water drops, Wind, Pouring water, Toilet flush, Thunderstorm, Baby crying, Sneezing, Clapping, Breathing, Coughing, Footsteps, Laughing, Tooth brushing, Snoring, Drinking/sipping, Knocking, Mouse click, Keyboard typing, Creaks(door/wood), Can opening,

Washing machine, Vacuum cleaner, Alarm clock, Ticking clock, Glass breaking, Helicopter, Chainsaw, Siren, Car horn, Engine, Train, Church bells, Airplane, Fireworks, Hand saw.

The Dataset is divided in 5 folds with 400 clips each.
We used folds *1,3,5* for *training* with a total of 1200 recordings (6000 patches); fold *2* for *validation* and fold *4* for *testing*, each containing 400 clips (2000 patches).

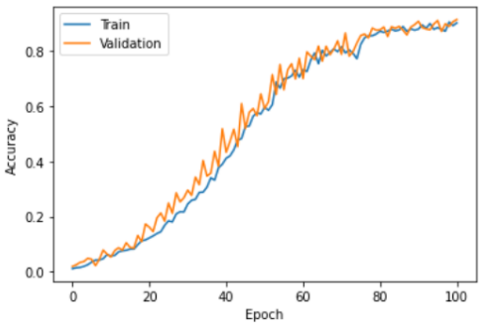


Figure 3.8: Train/Validation Accuracy on ESC50 Dataset.

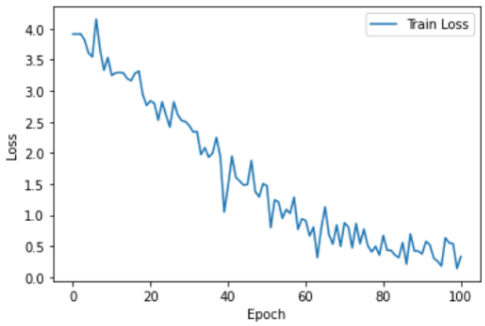


Figure 3.9: Train Loss on ESC50 Dataset.

CLASS LABEL	SUPPORT	PRECISION	RECALL	F1-SCORE	CLASS LABEL	SUPPORT	PRECISION	RECALL	F1-SCORE
Dog	40	1.00	0.75	0.86	**Footsteps**	40	1.00	0.90	0.95
Rooster	40	1.00	0.55	0.70	**Laughing**	40	1.00	0.87	0.93
Pig	40	1.00	1.00	1.00	**Brushing teeth**	40	1.00	0.92	0.96
Cow	40	1.00	0.85	0.92	**Snoring**	40	0.97	1.00	0.99
Frog	40	1.00	0.92	0.96	**Drinking sipping**	40	1.00	0.87	0.93
Cat	40	1.00	0.92	0.96	**Door wood knock**	40	0.89	0.62	0.73
Hen	40	1.00	1.00	1.00	**Mouse click**	40	1.00	0.95	0.97
Insects	40	1.00	0.92	0.96	**Keyboard typing**	40	1.00	0.95	0.97
Sheep	40	1.00	1.00	1.00	**Door wood creaks**	40	0.90	1.00	0.95
Crow	40	1.00	0.90	0.95	**Can opening**	40	0.91	0.80	0.85
Rain	40	1.00	1.00	1.00	**Washing machine**	40	1.00	0.80	0.89
Sea Waves	40	0.97	0.97	0.95	**Vacuum cleaner**	40	1.00	1.00	1.00
Crackling fire	40	1.00	1.00	1.00	**Clock alarm**	40	1.00	0.90	0.95
Crickets	40	0.97	1.00	0.98	**Clock tick**	40	0.95	1.00	0.97
Chirping birds	40	1.00	1.00	1.00	**Glass breaking**	40	0.19	0.97	0.33
Water drops	40	1.00	0.87	0.88	**Helicopter**	40	0.85	0.97	0.90
Wind	40	0.97	0.97	0.95	**Chainsaw**	40	1.00	1.00	1.00
Pouring water	40	1.00	0.87	0.93	**Siren**	40	1.00	1.00	1.00
Toilet flush	40	1.00	0.97	0.99	**Car horn**	40	0.96	0.67	0.80
Thunderstorm	40	0.90	1.00	0.95	**Engine**	40	1.00	0.92	0.96
Crying baby	40	1.00	1.00	1.00	**Train**	40	0.85	1.00	0.91
Sneezing	40	0.95	0.47	0.63	**Church bells**	40	1.00	1.00	1.00
Clapping	40	0.95	1.00	0.97	**Airplane**	40	1.00	0.92	0.96
Breathing	40	1.00	0.65	0.79	**Fireworks**	40	1.00	1.00	1.00
Coughing	40	1.00	0.65	0.79	**Hand saw**	40	1.00	0.92	0.96

Table 3.1: Classification Report for ESC50

As shown in Figure 3.8 and Figure 3.9 we reached a very good accuracy of 0.90 both on train and validation set, with a final training loss of 0.3.

	SUPPORT	PRECISION	RECALL	F1-SCORE
Accuracy		0.90	0.90	0.90
Macro Average	2000	0.96	0.90	0.92
Weighted Average	2000	0.96	0.90	0.92

Table 3.2: Accuracy and F1-Score Average for Test set on ESC50.

In Table 3.2 we can see that on the Test set our model can reach a value of *0.90* for accuracy and *0.92* value on F1-Score.
We can assert on the basis of these results that our classifier has very good performances when working with spectrogram patches as input.

3.4.2 UrbanSound8k Training

UrbanSound8k [17] is a Dataset made up of 8732 labeled sound with a duration less-equal to 4s.

The sounds belong to 10 different urban classes : air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, street music.

The Dataset is quite balanced with 1000 sound for all classes except 929 sound that belong to 'siren' class, 429 sounds from 'car horn' class and 374 sound that belong to 'gun shot' class.

The Dataset is divided into 10 fold, ready for a 10-Fold Cross Validation .
We have left the parameters described above unchanged during training, changing only the number of epochs.
First we have trained the classifier with a *10-Fold Cross Validation*, with only 2 epochs on every different fold, reaching an accuracy of *0.86*.
Than we used a total of 18449 patches for training set, 6193 patches for validation set and 6272 patches for test set. We ran the training for *15* epochs.

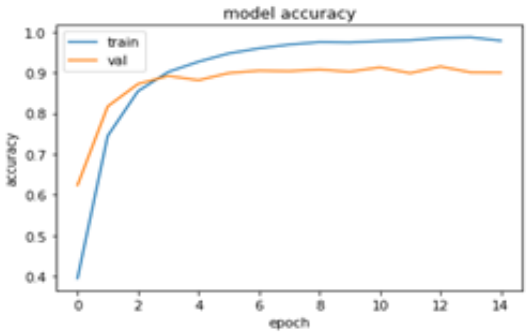


Figure 3.10: Train/Validation Accuracy on UrbanSound8k Dataset

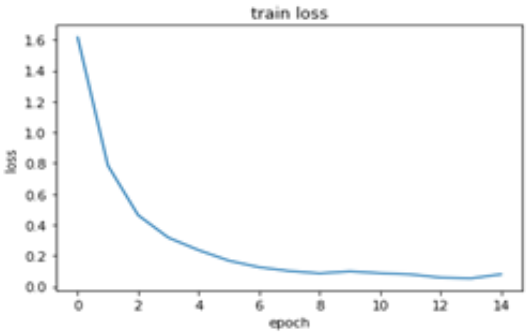


Figure 3.11: Train Loss on UrbanSound8k Dataset

We reach a best Validation accuracy of *0.91* and a best training loss of *0.05* as we can see in Figure 3.10 and Figure 3.11
Results on test set shows an accuracy of *0.88* and Average F1-Score equal to *0.89* as we can see in Table 3.4.

CLASS LABEL	SUPPORT	PRECISION	RECALL	F1-SCORE
Dog bark	700	0.93	0.96	0.95
Children playing	185	0.96	0.90	0.93
Car horn	740	0.68	0.89	0.77
Air conditioner	696	0.89	0.80	0.84
Street music	691	0.91	0.91	0.91
Gun shot	860	0.97	0.97	0.93
Siren	78	0.91	0.91	0.91
Engine idling	740	0.94	0.96	0.95
Jackhammer	710	0.86	0.97	0.91
Drilling	872	0.92	0.62	0.74

Table 3.3: Classification Report for US8k.

	SUPPORT	PRECISION	RECALL	F1-SCORE
Accuracy		0.88	0.88	0.88
Macro Average	6272	0.90	0.89	0.89
Weighted Average	6272	0.89	0.88	0.88

Table 3.4: Accuracy and F1-Score Average for Test set on US8k.

Thanks to the lower number of classes compared to ESC50 we are able also to

look at the *Confusion matrix* in Figure 3.12.

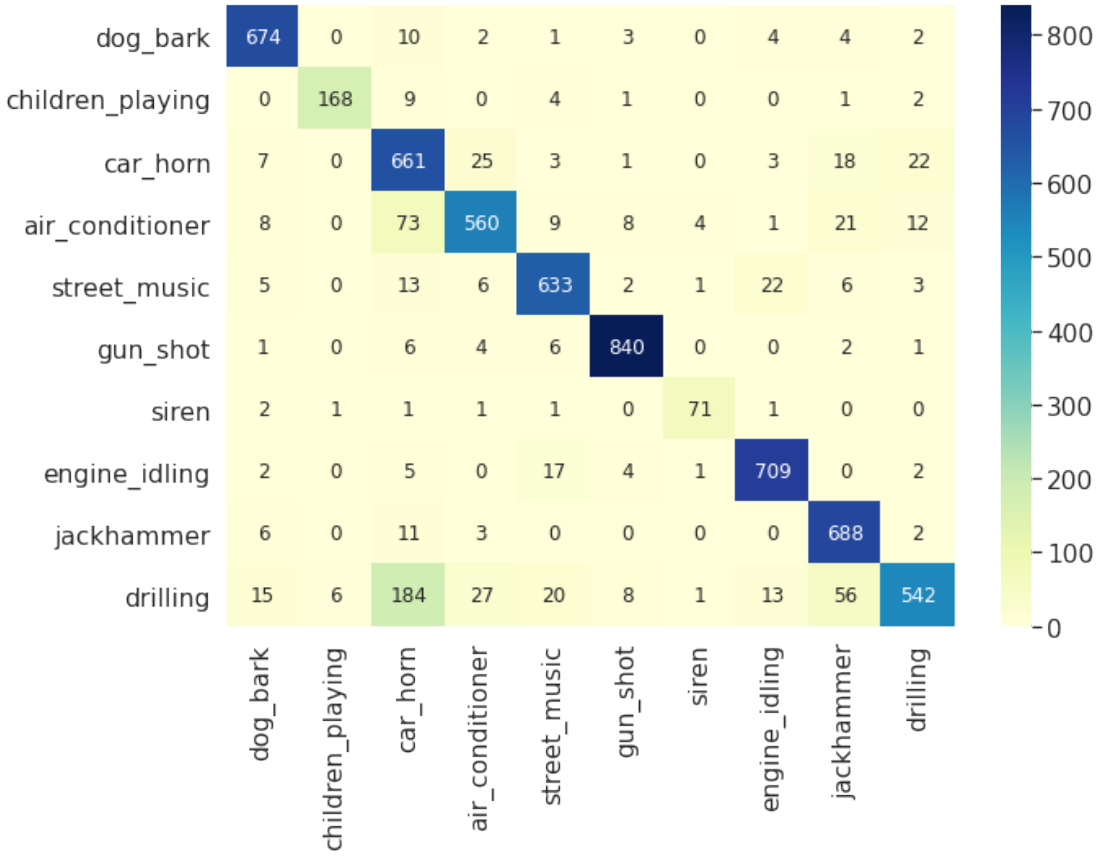


Figure 3.12: Confusion Matrix US8k Dataset

All the results point out again very good performance for audio classification. All classes are well recognized except for some confusion between classes 'car horn' and 'drilling'.

Chapter 4

Proposed Methodology

In this chapter will be described a general methodology to produce explanations for a black-box model and then will be introduced *A-EBAnO*, that is a specialized approach of the general methodology in the context of audio classification. The general methodology has the aim to produce explanation for image classification and it is already implemented.

Starting from the fact that the general methodology produces explanations for Neural Networks that deal with image classification and that the model we developed, described in the previous chapter, classifies audio by analyzing as input patches of the Log-Mel spectrogram, which are in fact images, we wondered if it was possible to produce explanations for audio classification, exploiting the internal algorithms of the general framework with appropriate and careful modifications.

The proposed technique is applied in the new context of audio classification and studied to see if we can achieve satisfactory results from the point of view of comprehensibility of *local explanations.*
We will present all the changes made to the framework to adapt to the new input type, paying particular attention to the new feature extraction techniques used in A-EBAnO for it to adapt to the new context.

4.1 EBAnO

EBAnO [18] is an innovative explanation framework able to produce explanations for the decision-making process of Deep Convolutional Neural Networks.

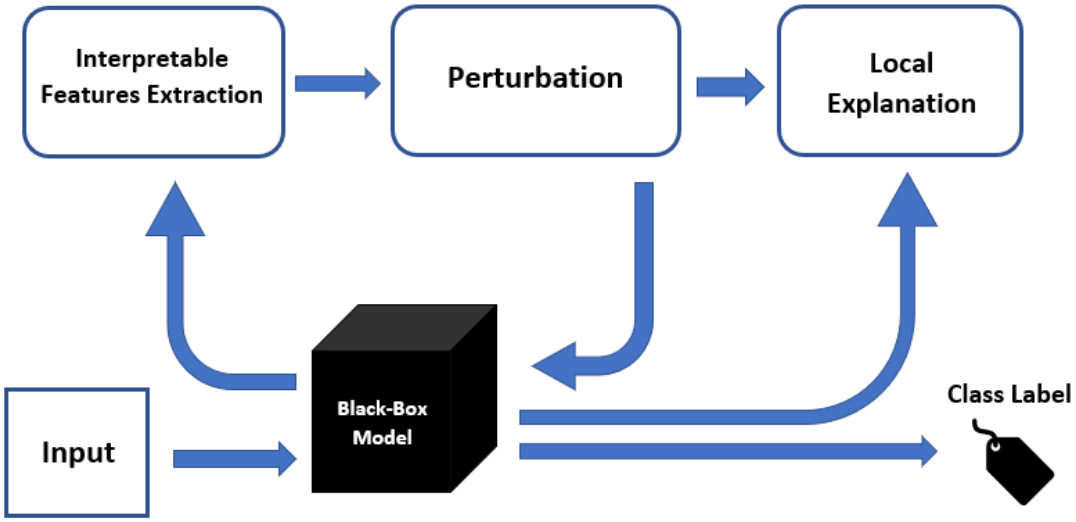


Figure 4.1: EBAnO Local Explanation process

The explanations are provided through an unsupervised extraction of the acquired knowledge of a black-box model contained in multiple convolutional layers simultaneously. The extracted knowledge is called *Interpretable Feature*.
The interpretable features are analyzed by means of perturbation in order to understand their *Influence* and Influence Precision on classification output, that it is finally visualized both with *visual* and *numerical* explanation. Figure 5.1 shows all the process.

4.1.1 EBAnO Input

EBAnO can receive in input unstructured data, such as images or text. The most consolidate part of the framework works with images as input. A-EBAnO work also with images but with a very important difference. While EBAnO produces explanations for the single image in input, A-EBAnO to produce the explanation for the classification of the audio in input has to group together all the explanations on the single Log-Mel spectrogram patches to have a single and complete explanation.

4.1.2 EBAnO Black-Box Models

EBAnO could work with all type of DCNNs. EBAnO for images classification has been tested with the most used *Convolutional Neural Network* as VGG (VGG16 and VGG19), Inception-v3 and Inception-ResNet.

In our context we used the model build upon VGGish, that is a modified version of VGG, as described in section 3.3.

4.1.3 EBAnO Interpretable Features Extraction

The extraction of *interpretable features* is the most important and crucial phase of the entire process. During this phase the framework must be able to identify significant parts of the input that are responsible for classification.
In the context of EBAnO for images the image input is composed by pixel, but an analysis on every single pixel leads only to results not interpretable for a human being.
The objective of the extraction becomes therefore that of identifying significant portions of the input, formed, for example, by pixels correlated with each other.
In order to do this EBAnO utilizes an unsupervised clustering analysis of *hypercolums*. Every pixel on input image is characterized by a vector, called hypercolumn, of activations of all the layers of DCNN above that pixel [19] and describe all the acquired knowledge of the model about that pixel.
Proceeding with clustering analysis of hypercolums, EBAnO can put together correlated pixels and find the portion of the input they belong to. In this way the clustering is guided only by the weights of the Neural Network that are stored in the hypercolumn, exploiting the inner knowledge of the model to produce trustfull and fitting local explanations.
The algorithm used for the clustering is *K-Means*. The number of 'means' that we use during the clustering determines the number of portions in which the input will come subdivided.

In our context we decided to use this type of feature extraction but we developed two other feature extraction techniques in order to better analyze the properties and characteristics of our different domain of the input and also we adapted the feature extraction technique in order to have a clustering on the hypercolums of all the patches of the Log-Mel Spectrogram and thus indirectly on the complete spectrogram itself.

4.1.4 EBAnO Perturbation

In order to understand the the influence of input data on the prediction result, the original input data is iteratively *perturbed* at each feature calculated in the previous step and a new prediction is made on the modified input.
Comparing the results of the two different predictions, one on the original input and the other on the perturbed input, three possible outcomes are possible :

Original Prediction equal to Perturbed Prediction In this case the modification of the input has not brought any impact in the prediction and this indicates that the concept represented by the feature *does not have any relevance* for the predicted class.

Original Prediction weaker than Perturbed Prediction This indicates that the feature taken in analysis has a *negative impact* in the prediction, therefore modifying or eliminating the concept expressed by the feature, the Neural Network is more accurate in the prediction of the correct class to which the input belongs.

Original Prediction stronger than Perturbed Prediction Observing a higher uncertainty in the prediction of the correct class after applying the perturbation, we can understand that the concept expressed by the feature is important for the prediction because it has a *positive influence* that is mitigated if the feature undergoes some kind of modification. The nature of the perturbation is closely related to the type of input data. In the context of EBAnO used on Image Classification task the perturbation used is Gaussian Blur which is suitable for the images domain.

The inputs of our model are, in the same way, images that come from a visual representation of a type of data that belongs to a completely different domain, that of audio. For this reason it was necessary to completely change the type of perturbation choosing one more suitable and consistent for our context.

4.1.5 EBAnO Numerical and Visual Local Explanations

In order to have the possibility to produce a local explanation, that presents both a visual and a numerical part and that shows clearly the influence of each interpretable feature on a particular class, two indices are proposed:

- nPIR : normalized Perturbation Influence Relation
- nPIRP : normalized Perturbation Influence Relation Precision

normalized Perturbation Influence Relation nPIR is a numerical representation of the influence of the analyzed feature on the prediction of the class we are interested in, taking in account original prediction and perturbed prediction. The influence can be positive, neutral or negative as described in subsection 4.1.4
In mathematical terms, given :

- C : set of class of interest
- $ci \in C$: class of interest
- F : set of interpretable feature
- $f \in F$: analyzed feature
- $p_{o,ci}$: probability of original input to be classified with the class of interest
- $p_{f,ci}$: probability of the same input to be classified with the same class of interest after f perturbation

Firstly is calculated how much the perturbation on f *impact* the probability :

$$\Delta I_f = p_{o,ci} - p_{f,ci} \tag{4.1}$$

It's domain is [-1,1] since domain of probabilities is [0,1].
If ΔI_f is positive this means that the original probability $p_{o,ci}$ is greater than $p_{f,ci}$, probability with perturbed feature and so the feature has a *positive* impact on the prediction of the class of interest.
Otherwise if $p_{o,ci}$ is lower than $p_{f,ci}$, ΔI_f is negative and this mean a *negative* impact of the feature on the prediction.
If ΔI_f is equal to zero than probabilities are the same and the feature is *neutral* on the prediction process.

To calculate the relative impact of the perturbed feature f another index called *Symmetric Relative Influence* is introduced:

$$SRI_f = \frac{p_{o,ci}}{p_{f,ci}} + \frac{p_{f,ci}}{p_{o,ci}} \tag{4.2}$$

Both contributions, (5.1) and (5.2), are combined together defining *Perturbation*

Influence Relation :

$$\begin{aligned} PIR_f &= \Delta I_f * SRI_f \\ &= (p_{o,ci} - p_{f,ci}) * (\frac{p_{o,ci}}{p_{f,ci}} + \frac{p_{f,ci}}{p_{o,ci}}) \\ &= p_{f,ci} * (1 - \frac{p_{f,ci}}{p_{o,ci}}) - p_{o,ci} * (1 - \frac{p_{o,ci}}{p_{f,ci}}) \end{aligned} \tag{4.3}$$

Finally *PIR* is normalized in range [-1,1] obtaing the *normalized Perturbation Influence Relation* :

$$nPIR = softsign(PIR) \tag{4.4}$$

where

$$softsign(x) = \frac{x}{1 + |x|}$$

When *nPIR* has a value close to 1 , feature f has very important positive influence for the prediction of the class of interest, if it is 0 feature is neutral and if *nPIR* is near to -1 the feature under analysis has an important negative impact on classification.

normalized Perturbation Influence Relation Precision nPIRP reflects how much the influence of a feature f is focused on the class of interest. In a multi-class problem a particular feature perturbation can impact more than one class and more are the classes influenced by that feature less *precise* is the contribution of the feature on classification.
In mathematical terms:

$$\xi_{ci} = p_{o,ci} * |nPIR_{ci}| \tag{4.5}$$

$$\xi_{C\backslash ci} = \sum_{c}^{C\backslash ci} p_{o,c} * max(0, nPIR_c) \tag{4.6}$$

Equation (5.5) indicates the absolute influence of feature f over class of interest.
Equation (5.6) indicates the sum of only positive influence on all the classes except the class of interest.
Both contributes are weighted by the probability of original input to belong to each

different class. The *Perturbation Influence Relation Precision* has the following formulation :

$$\begin{aligned} PIRP_f &= \Delta I_f(\xi_{ci}, \xi_{C\setminus ci}) * SRI_f(\xi_{ci}, \xi_{C\setminus ci}) \\ &= \xi_{C\setminus ci} * (1 - \frac{\xi_{C\setminus ci}}{\xi_{ci}}) - \xi_{ci} * (1 - \frac{\xi_{ci}}{\xi_{C\setminus ci}}) \end{aligned} \tag{4.7}$$

PIRP is normalized in range [-1,1] obtaing the *normalized Perturbation Influence Relation Precision* :

$$nPIRP = softsign(PIRP) \tag{4.8}$$

If the *nPIRP* has a value close to 1 is very focused on the classification of the class of interest while if it has a value near -1 the impact of feature f on the class of interest is lower respect the impact on other classes. Finally a value of *nPIRP* equal to 0 means that f has the same influence on the class of interest and on other classes.

nPIR and *nPIRP* don't take in consideration input type, model architecture and task so they represent a very general approach that we decided to leave unchanged in our approach.

Local Explanations An explanation e is composed by a set of features that are separately perturbed and for each one feature f in this set *nPIR* and *nPIRP* indexes are calculated.
EBAnO produces explanations, extracting and analyzing from two to a maximum number of features, that we can set, and among all the produced explanation EBAno select the *most informative* one. The best explanation is characterized by the best score S , where S is defined as :

$$S(e) = max((nPIR_f(e)) - min(nPIR_f(e)) \tag{4.9}$$

Numerical Explanation is a graph showing *nPIR* and *nPIRP* for every feature f composing the explanation.

Visual Explanation is a visual representation of the interpretable features extracted, in which the contribution of each feature is represented by a color scale based on the nPIR value directly on the input data.

4.2 A-EBAnO

In this section are described general workflow and architecture of *A-EBAnO* to produce *local explanations* of the predicted class label of an input audio. After preprocessing of the input audio, it is feeded into the classifier, *interpretable features* are extracted from the input exploiting the *inner knowledge* that the black-box model has learned during the training phase, iteratively every feature, among all those that compose each explanation, are *perturbed* and the best local explanation is computed analyzing the differences between the prediction probabilities before and after perturbation, producing a numerical and visual explanation of the classification of the input audio.

A-EBAnO stands for *Audio-EBAnO* to to emphasize how the work is a specialization in the *audio classification* task of the EBAnO general framework.

4.2.1 A-EBAnO Input Pre-Processing

A-EBAno receives in input an audio file in *Waveform Audio File* format, commonly referenced as *'.wav'*. It represent the principal format for raw uncompressed audio, used to store high quality audio files.

For the *audio classification* is used its *Log-Mel spectrogram* from which the *interpretable features* are extracted.

The input '.wav' file is converted to *mono*, an array of only one dimension, it is *resampled* to 16000 Hz (16 kHz) and the Log-Mel spectrogram is computed.

Given :

- *Input audio data* : the mono dimensional array of waveform data
- *Audio Sample Rate* = 16 kHz
- *Log-offset* = 0.01 : a value to add when Logarithm is calculated in order to avoid -Inf.
- *Window Length seconds* = 25 ms : durationof each windows to analyze
- *Hop Length seconds* = 10 ms : advance between successive windows

the input is divided in sequence of overlapping frames with :

- *Window Length* = *Audio Sample Rate* * *Window Length seconds* = 400 samples in each frame.

when each frame start

- *Hop Length = Audio Sample Rate * Hop Length seconds* = 160 points after the proceeding one.

Each frame is *windowed* with a 'periodic' *'Hann Window'*, a *cosine* of period equal to *Window Length*, and the *Short-Time Fourier Transform* magnitudes are calculated using:

- $FFT\ Length = 2 * \lceil \frac{\log(WindowLenght)}{\log(2)} \rceil = 512$

building a two dimensional array where each row contains the magnitudes of the $\frac{FFTLenght}{2} + 1 = 257$ values of FFT for the corresponding frame of input.

The matrix S of STFT magnitudes arranged as *frames x bins* is post multiplied to a matrix A of *Mel Weights* to build a *mel spectrogram* $M = S * A$ with shape *frames x number of mel bins*.
Matrix A is constructed using:

- *Number of mel bins* = 64 : number of column of S representing how many bands are present in the resulting Mel Spectrum
- *Number of spectrogram bins* = 257 : bins contained in the source spectrogram
- *Audio Sample Rate* = 16 kHz
- *Lower Edge Hertz* = 125 Hz : lower bound of frequencies to be included in the Mel spectrum
- *Upper Edge Hertz* = 7500 Hz : upper bound of frequencies to be included in the Mel spectrum

The *Log-Mel Spectrogram* is finally computed as log(M + *Log-Offset*).
At the end the Log Mel spectrogram is divided into patches that contains :

- *96* frames of 10 ms duration each.
- *64* frequency bands.

From every patch separately the *interpretable features* are extracted, they pass through the perturbation phase and the best *explanation* of the prediction of the class label for that patch is produced.

In order to have the explanation on the complete Log-Mel spectrogram , A-EBAnO loop over all the patches of the input spectrogram, doing all computations and at the end combine individual explanations to produce the full local explanation.

4.2.2 A-EBAnO Black-Box Model

In order to produce *explanations* of the classification result of an input, we need a Deep Neural Network that takes in input the audio file and makes the prediction. A-EBAnO in the current implementations uses the network builded upon *VGGish* and described in section 3.3.

4.2.3 A-EBAnO Features Extraction

Features extraction is a very critical phase of the overall workflow of A-EBAnO. It is responsible to find not only *usefull* but in particular easily *interpretable* parts of the input that have important impact on the prediction of the class label.

In our work we firstly have conducted experiments with the features extraction technique used on the general framework. In our context we refer to this technique as *Single Patch Features Extraction*

Analyzing its weaknesses when used in our context we adapted this technique for be suitable in our context and we refer to it as *Spectrogram Features Extraction*

Audio has a fundamental characteristic which is that of being a *signal* that varies over *time* and the Log Mel Spectrogram is a visual representation of the audio's amplitude while it varies over time at different *frequencies*.
In order to have explanations that which emphasize whether and how the neural network exploits this two fundamental characteristics to produce class prediction, we build up two new features extraction techniques in which the input is divided once in *frequency bands* and the other in *time bands*. We refer to these as *Frequency Bands Features Extraction* and *Time Bands Features Extraction*

To summarize we experiments with 4 different features extraction techniques:

1. *Single Patch Features Extraction*
2. *Spectrogram Features Extraction*
3. *Frequency Bands Features Extraction*
4. *Time Bands Features Extraction*

Single Patch Features Extraction We first investigated the behavior of the feature extraction technique of the general framework in our new context to see if it had the capabilities of extracting portions of images that could be easily analyzed

and that it could select portions of the input that had some significance for the classification of the class label of the audio input.
This features extraction technique analyzes every single patch singularly, studying it as an input in itself not taking in consideration the fact that it is only a portion of the complete spectrogram to analyze.
Weights on the all Convolutional Layers that we want to study are extracted and for every unit the *hypercolumn* is calculated . Than hypercolumn vector dimensionality is reduced through *Principal Component Analysis* (PCA).

PCA is the algorithm to calculate *principal components* of some input data points in order to do a change of basis of the data , projecting input only on the first few principal component, reducing in this way the dimensionality of the input. *Principal components* of a series of input data points in a real *n-space* is a collection of *n direction vectors* where every vector is the direction of a line that best fit input points, i.e. for which the *average squared distance* between the points and the line is minimized while being *orthogonal* respect to other direction vectors. The first principal component is the direction that *maximize variance* of the projected data. Experimenting over 200 audio input we find that the number of principal components that maximize *cumulative explained variance* is 65.

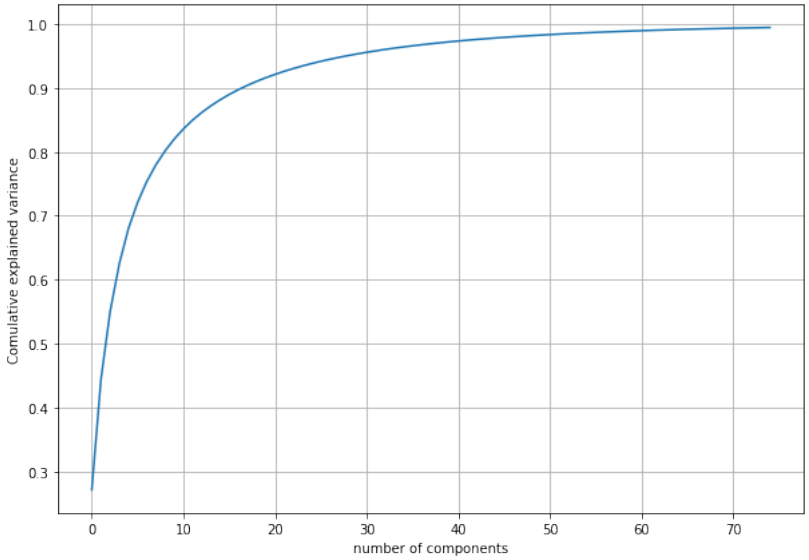


Figure 4.2: Cumulative explained variance w.r.t number of principal components.

A clustering on the reduced hypercolums vector is done by the *K-Means* algorithm and a *feature map* is produced with number of feature spacing from minimum 2 features to a maximum number of features that we can set as parameter of

A-EBAnO.

Algorithmicaly (Figure 4.3) the features map is a 2 dimensional array of the same size of the input patch where every cell in the matrix, representing a pixel of the Log-Mel patch, contains the *id* of the cluster that it belong to and it is visualized using different colors for different clusters (Figure 4.4).

1	1	1	1	2	1	2	2	1	2
1	1	1	1	2	1	2	2	1	2
1	1	1	1	2	1	2	2	1	2
1	1	1	1	2	2	2	2	1	2
1	1	1	1	2	2	3	3	1	2
2	2	2	2	2	2	3	3	1	2
4	4	4	4	2	3	3	3	3	2
4	4	4	4	2	3	3	3	3	2

Figure 4.3: Example of internal representation of a low resolution *features map* with 4 features.

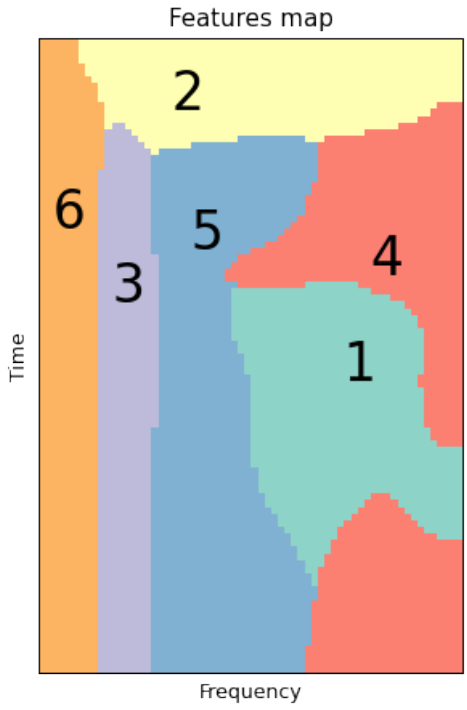


Figure 4.4: Visual representation of *Single patch features map* with 6 features.

Spectrogram Features Extraction With the previous technique we have seen how the knowledge learned by the black-box model is mined by A-EBAnO in order to build explanations for the predicted class analyzing one patch at time of the complete Log-Mel spectrogram.

The main weakness of this approach is that we lose sight of the fact that the patch belongs to a single input, i.e. the complete audio Log-Mel spectrogram, and that therefore the patches are closely related to each other, being together the union of the spectrogram. Taken individually, the extraction of features from each patch could lead to the loss of information and parts of the input that impact the classification, especially between the points of junction between the different patches.

The solution that we propose is to loop a first time trough all the patches that make up the Log-Mel spectrogram, calculating the hypercolumns for each patch and *concatenating* them into a data structure.
After collecting all the patches, the data structure contains the hypercolums of all the patches at the same time and so indirectly contains the knowledge acquired by the model on the *complete spectrogram*.
We proceed by calculating on this data structure the clustering of the hypercolums and we obtain the *interpretable features* as if we had in input the total Log Mel Spectrogram.
After extracting the features the data structure is framed with patches of the same size of the input patches and the framed data structure is stored into A-EBAnO.
In A-EBAnO workflow another loop start from the first patch in order to produce the explanation on the entire spectrogram and using this features extraction technique every patch have already calculated its *features map* to which it is associated through its *index* in the loop. For example patch number 2 finds its feature map at index 2 of the data structure that collects the complete feature map on the spectrogram.

In our implementations the data structure is a list, so the complete Log-Mel spectrogram is stored into a list of 2-dimensional array.

With this different features extraction technique the aim is to recover the uniqueness of the single patches in the complete spectrogram to try to have features with a more *continuous trend* along the whole spectrogram and may be less clearly separated passing from one patch to the next, as they appear using the first feature extraction technique.

Algorithm 1 Spectrogram Features Extraction. Given n number of patches

▷ Initialization
$HC \leftarrow empty$ ▷ Data structure to store hypercolums of each patch
▷ Execution
for i = 0 to n **do**
    $p \leftarrow load\_patch(i)$ ▷ Load current patch
    $hc\_patch \leftarrow extract\_hypercolumns(p)$ ▷ Extract hypercolumns of current patch
    $HC[i] \leftarrow hc\_patch$ ▷ Store current patch hypercolumns
end for
$hc \leftarrow concatenate(HC, axis = 0)$ ▷ Get a single data structure
$features\_map \leftarrow clustering(hc)$ ▷ Build the features map
$framed\_map \leftarrow frame(features\_map)$ ▷ Divide in frame the complete features map
▷ A-EBAnO workflow continue and every patch at index i find the correspondent features map on *framed_map[i]*

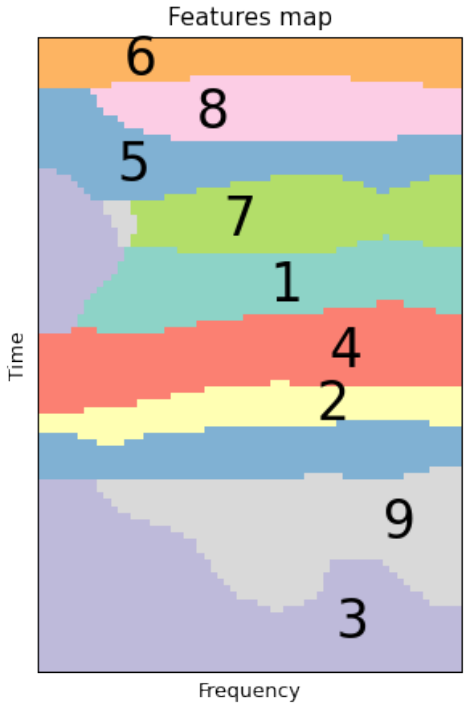


Figure 4.5: Visual representation of patch features map with 9 features calculated using *Spectrogram Features Extraction* technique.

The objective of this and previous features extraction techniques is to extract impacting portions of the input for the classification of the predicted class.
These techniques helps us to understand what parts of the spectrogram are more important for the black-box model and if it is correctly focused and can follow the portions of the spectrogram that present *sound energy* patterns consistent to the input sounds.

The two features extraction methods are useful also to have visualization of the general approach of the model in the input classification.
To be practical, observing Figure 4.4 it is possible to recognized a clustering that is focused on creating *vertical* regions while Figure 4.5 present a division in *horizontal* features.
In the next sections will be clear that the first figure show that the model is more focused on the *frequencies* of the input patch while the second figure show that the best features which to build the explanation are taken focusing on a *time* division of the input.

Frequency Bands Features Extraction The Log-Mel spectrogram show the signal strength at the various *frequencies*.
In order to focusing the explanation on this important characteristic of the spectrogram and to be able to strictly analyze how the frequency of the input audio impact the outcome prediction , we build up this features extraction technique.

The objective of the features extraction is to create a clustering of input points correlated each other. The analysis of *frequency bands* really don't need to mine the inner knowledge of the model in order to calculate the feature map. Instead we directly create the *features map* algorithmically.

As we can see in figures on the previous section and discussed in subsection 4.1.1, every patch has a 96x64 dimension, where 96 stands for 96 frames of 10ms each and 64 stands for 64 Mel frequency bands.
This means that visually every patch is represented with the time on the y-axis and frequencies on x-axis.
The *frequency bands* feature map is represented as a set of *vertical* clusters for this reason.

In the implementation of the technique we could choose between two different ways of working:

Division into static frequency bands Using this approach we can decide to divide every patch with a chosen number of features, disregarding the maximum

number of feature chosen at the start of the analysis.

Division into dynamic frequency bands A-EBAno build an explanation using from 2 to the max number of features chosen as parameter , so the final number of bands composing the features map can change reflecting the number of features composing the most informative explanation for the patch. In this way consecutive patches could have a different number of features in the features map.

We chose to use the second strategy for a very important reason. An audio can vary a lot from one instant of time to another. Every patch represent approximately 1 second (960 ms) of the input audio and different patches can represent very different characteristic of the input respect to other patches, so we left to A-EBAnO to choose what are the right number of bands, for every different patch, to use in order to create the most informative explanation, so as not to risk losing sight of important information, as could happen using a static frequency bands division.

The objective of this features extraction technique is to highlights the impact of a certain frequency band on the classification of the class of interest. We can understand if the sound recognized in the audio file is classified using an appropriate range of frequencies.
A sound that we perceive as *low* is generated by a sound wave with a *low frequency* while a *high-pitched* sound is generated by a *high frequency* sound wave. If in the classification of a *low* sound, *high frequencies* have a more positive impact on the classification than there could be some kind of error or imprecision in the model, that should instead be based more on the *low frequencies*. The opposite is true for *high* sounds.

$$
\begin{matrix}
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 \\
1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4
\end{matrix}
$$

Figure 4.6: Example of internal representation of a low resolution *Frequency bands features map* with 4 features.

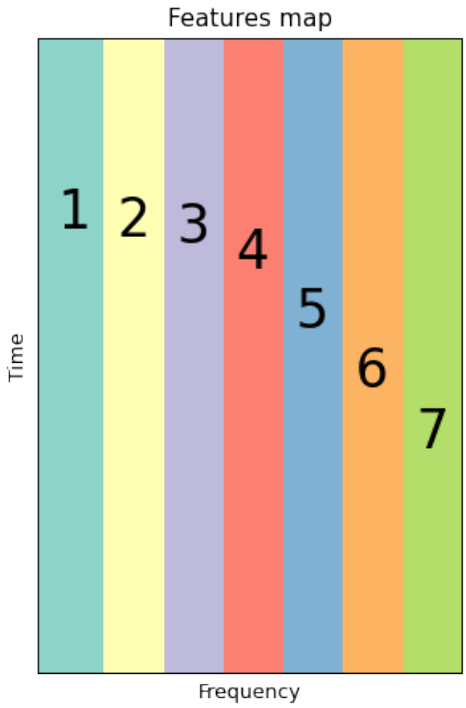


Figure 4.7: Visual representation of single patch *Frequency bands* features map with 7 features.

Time Bands Features Extraction Log-Mel spectrogram does not only show the signal *strength* at the various frequencies but also how it change *over time*. Each instant of time of the input audio can represent an important event for classification both in terms of positive and negative impact. At any instant there may be a change in sound or its immediate absence. We therefore thought that we needed a tool that would make us able to analyze the input by *time bands*. All the analysis done for the previous feature extraction technique also applies to this one. The *time* on the Log-Mel spectrogram patches is located on the y-axis so the *time bands* feature map is represented as a set of *horizontal* clusters.
We also thought that also for this technique the best solution was to have a *dynamic* time bands division.

The aim of this technique is to help us to understand if the black-box model can distinguish exact moment of time that are important for the classification.
If at a certain instant of time the sound that is present, clearly don't belong to the input audio class or if there is not sound at all and the classification gave an *exact* result for the input audio classification, we expect the explanation to show how that exact instant of time has zero or negative impact.

If this is not the case then some kind of problem could be present in the classifier.

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4	4	4

Figure 4.8: Example of internal representation of a low resolution *Time bands features map* with 4 features.

Figure 4.9: Visual representation of single patch *Time bands* features map with 8 features.

4.2.4 A-EBAnO Perturbation

After extracting *interpretable features*, A-EBAnO proceeds with an iterative *perturbation* against all these features producing a set perturbed Log-Mel spectrogram patches, where portion that not belong to the feature under analysis are left unchanged while the input portion that belong to the feature that A-EBAno is studying are perturbed . The perturbed patches are fed again into the DNN under analysis and the prediction of the perturbed input is made. By comparing the prediction on the original input and on the perturbed one, the *influence* and *influence precision* are calculated.
To obtain the perturbed patches we had two solutions:

Perturbation on the input audio When A-EBAnO workflows start , we can perturb the input audio, calculate and store its Log-Mel spectrogram patches into A-EBAnO. During the production of the perturbed patches we only need to retrieve the exact patch of the spectrogram computed on the modified input and combine it with the original spectrogram patch under analysis.

Perturbation on the spectrogram patch We directly modify the input patch with some consistent *noise* only on the portion of the patch that belong to the current feature. In the current implementation of A-EBAnO we decide to use the second strategy because is less computationally expensive and faster and also because the real input of the classifier is not the audio but its Log-Mel spectrogram so we think that is more appropriate to modify directly the patch.

In this work we propose as perturbation *Additive White Gaussian Noise* (AWGN). It is *Gaussian* because he have a normal distribution, *white* because it has uniform power on all the frequency band and *additive* because is added to the intrinsic noise that can be present in the perturbed signal.

Given the original patch and the feature map, every time a feature must be analysed, a *feature mask* is created where are highlighted only the points that belong to that feature. These points are marked with "1".
Than we build the *AWGN* using as *mean* of the distribution zero and as *standard deviation* the standard deviation of the audio power levels in the patch.
Than we use the feature mask to build the perturbed patch: points that not belong to the feature remain equals to themselves while the noise is *added* to the value of the points that are part of the interpretable feature.

Algorithm 2 Algorithm to produce a perturbed patch

procedure GET_PERTURBED_INPUT(i_p, f_m)
    ▷ i_p is input patch
    ▷ f_m is feature mask
    ▷ Execution
    $mean \leftarrow 0$
    $std \leftarrow std(i\_p)$ ▷ Standard Deviation
    $noise \leftarrow normal(mean, std)$
    $noisy\_patch \leftarrow i\_p + noise$
    $opposite\_mask \leftarrow 1 - f\_m$ ▷ mask for not-perturbed input point
    $p1 \leftarrow noisy\_patch * f\_m$ ▷ get only perturbed points of the feature
    $p2 \leftarrow i\_p * opposite\_mask$ ▷ get non perturbed points
    $pp \leftarrow p1 + p2$ ▷ combine together into perturbed patch
    return pp ▷ Perturbed patch is returned
end procedure

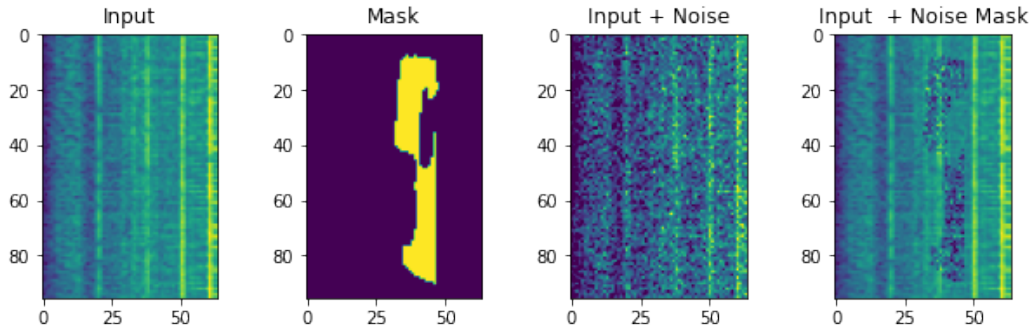


Figure 4.10: Perturbation process using *Spectrogram Features Extraction*. Perturbation using *Single Patch Features Extraction* is equal and is not reported.

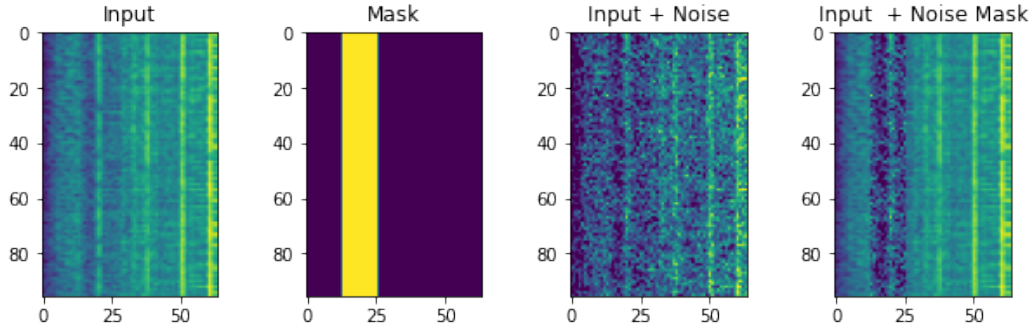


Figure 4.11: Perturbation process using *Frequency Bands Features Extraction*

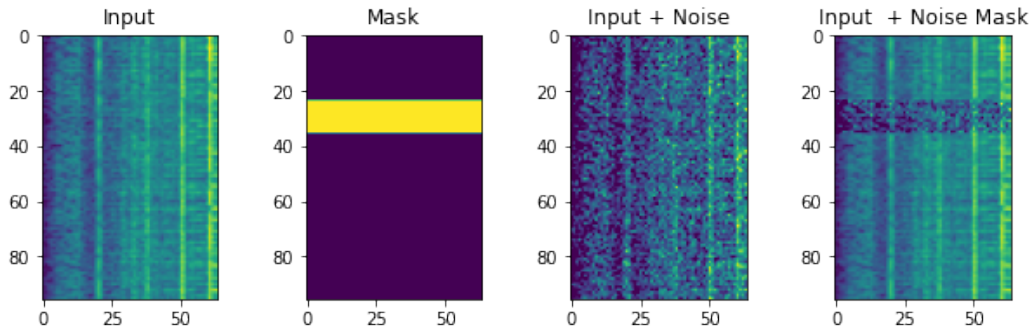


Figure 4.12: Perturbation process using Time Bands Features Extraction

4.2.5 A-EBAnO Post-Processing

An explanation to be usefull must must *clearly understandable* by all types of users. When we work with audio signals we are used to having *time* represented on the x-axis and others characteristic on the y-axis.
In our work we build explanations on audio classification based on the Log-Mel spectrogram of the input audio . It is a visual representation of the power of the sound present in the audio as it change *over time* at various *frequencies*. Working with spectrogram than we expect to have a visualization that uses the x-axis to represent time and y-axis to represent frequencies.

As we can see in all examples in the previous sections, A-EBAnO takes the audio input, calculate its Log-Mel Spectrogram and then divide it in patches of shape 96x64. Visualizing the patches, and consequently all the workflow step outputs of A-EBAnO, we can see that the two axis, time and frequency, are swapped. It is possible to leave this convention for axes unchanged, but the output explanation would be *difficult to understand* and study.

In order to present a clearly understandable explanation, after all calculations is done and before producing the visual representation, we proceed to post-process the input patches and the output visual explanation.
Comparing the Log-Mel spectrogram calculated with A-EBAnO with Log-Mel Spectrogram calculated with Librosa [20], a Python package for audio analysis, we understand that, in order to have a coherent visual representation of the spectrogram, we have to both *rotate* and *flip vertically* it.

Post-Processing is applied to every patch of the *features map* and consequently to every patch composing the *visual explanation* and then inputs with the right orientation are combined together in the final explanation.

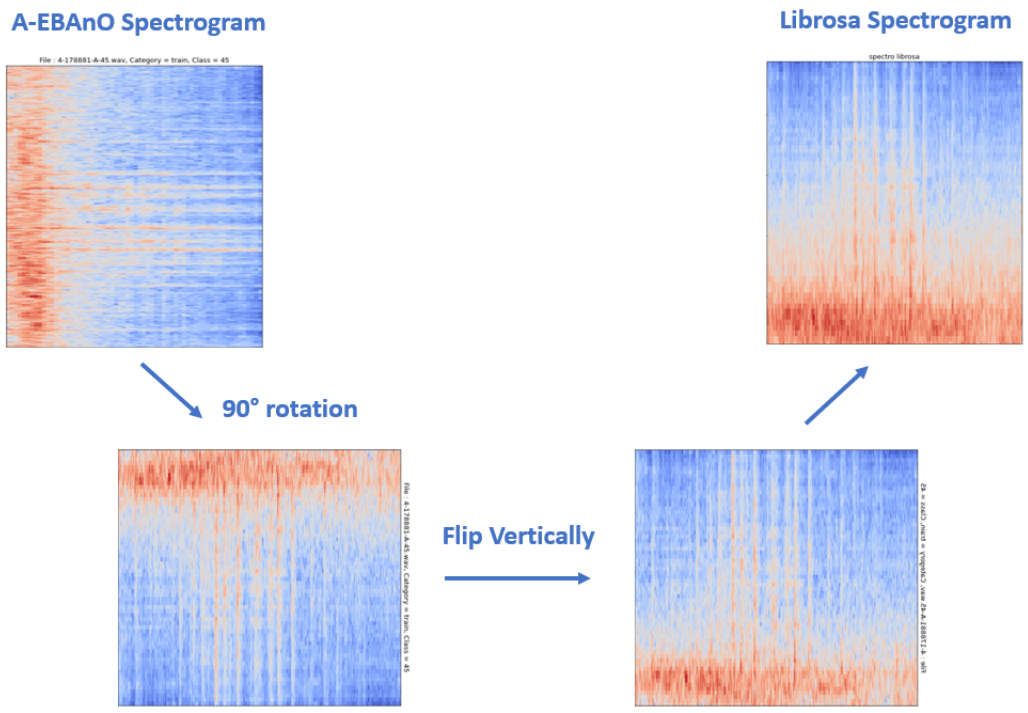


Figure 4.13: Visual post-processing visualized on the complete spectrogram

4.2.6 A-EBAnO Visual and Numerical Local Explanations

Visual and numerical local explanations are the final outputs of A-EBAnO. Their objective is to show in two different way the influence of every extracted interpretable feature on the classification of the audio input.

Visual explanation is composed by two different contributes:

Interpretable Features Map It is a visual representation of the extracted features in which each one is marked with a numerical id and a different color.

Visual Explanation Each interpretable feature impact on the classification is shown using a graduated color scale from green, positive influence, to red, negative influence, passing through gray, neutral influence, directly on the Log-Mel spectrogram. Each explanation specify the label of the class of interest and what features extraction technique has been used coded with a numerical id : 0 for *Single*

Patch Features Extraction, 1 for *Frequency Bands Features Extraction*, 2 for *Time Bands Features Extraction*, 3 for *Spectrogram Features Extraction*.
On the abscissa time is reported while on the ordinate is reported frequency.
The indexing of the features is not incremental but for every patch start again from 1 to the maximum number of extracted features. In this way we think the explanation is more easily readable because the ids would tend to reach high numbers creating confusion on the visualization.
The following figures shows features map and visual explanation on the same audio input Log-Mel spectrogram for the 4 different feature extraction techniques.

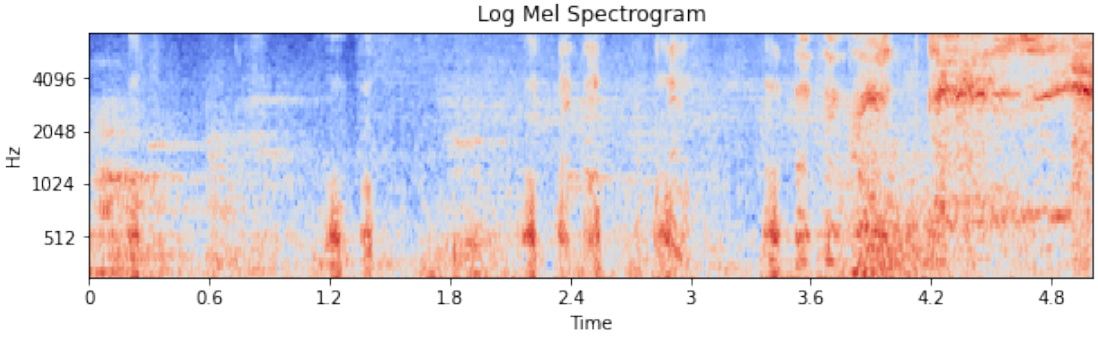


Figure 4.14: Audio Input Log-Mel spectrogram

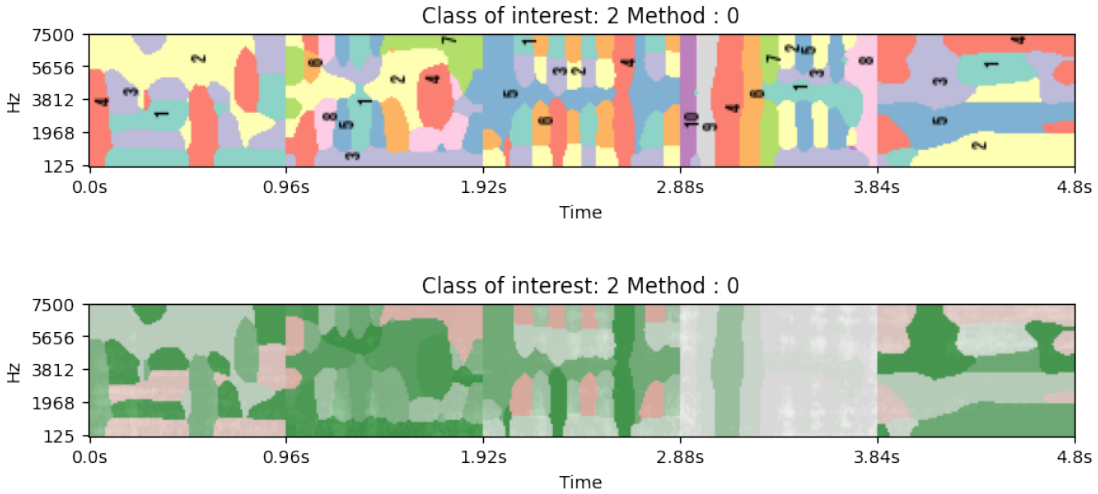


Figure 4.15: Features map (top) and visual explanation (down) using *Single Patch Features Extraction* technique.

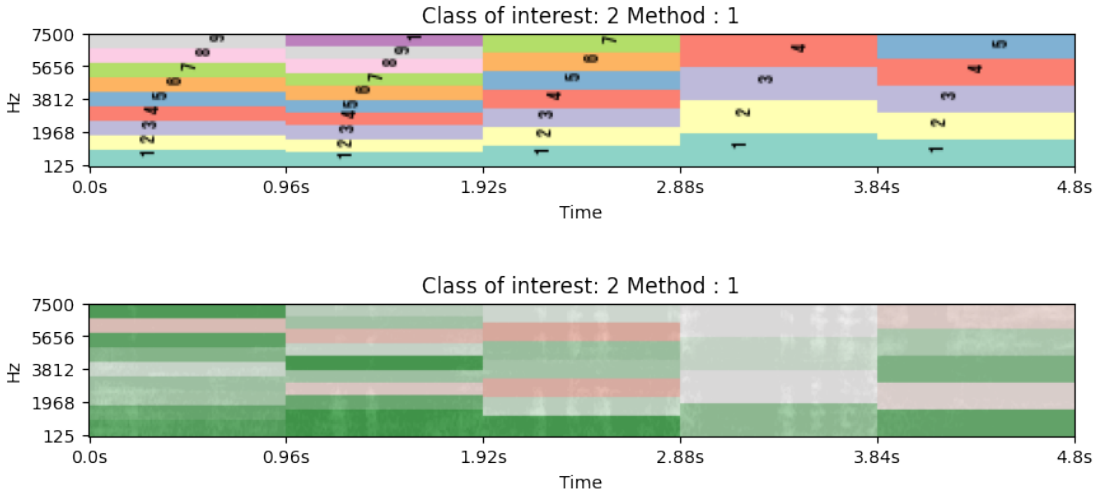


Figure 4.16: Features map (top) and visual explanation (down) using *Frequency Bands Features Extraction* technique.

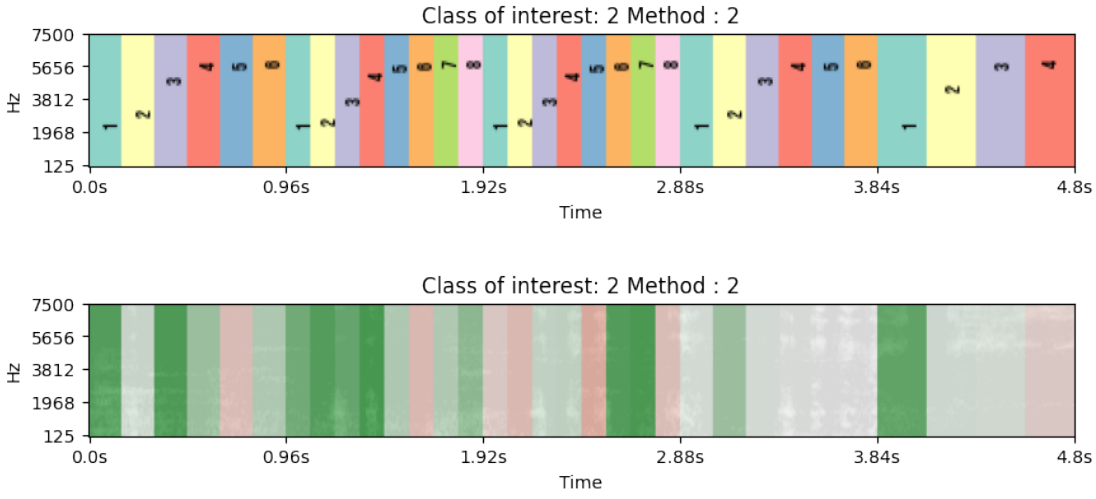


Figure 4.17: Features map (top) and visual explanation (down) using *Time Bands Features Extraction* technique.

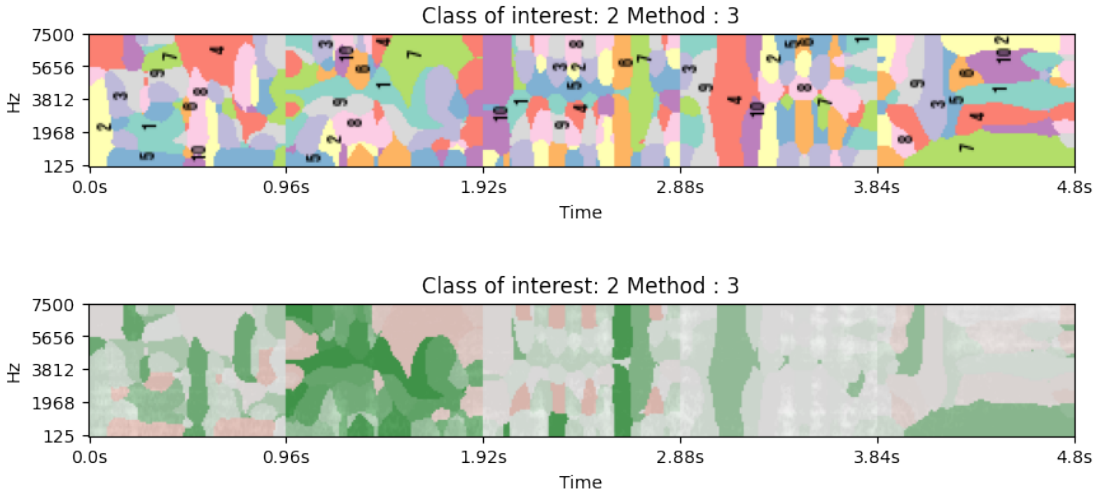


Figure 4.18: Features map (top) and visual explanation (down) using *Spectrogram Features Extraction* technique.

The objective of *local Numerical Explanation* is to show in a easy way the two proposed index values for every different interpretable feature extracted.

Numerical explanation shows with a *bar chart* the value of *nPIR* reached by every features. nPIR stand for *normalized Perturbation Influence Relation* and it range from -1 to 1, where -1 indicates a very negative influence of the feature on the classification result while 1 indicates a very positive influence of the feature on the classification of the audio input.
The second index ,*normalized Perturbation Influence Relation Precision* (*nPIRP*) , is showed as a mean value over all the patches composing the complete Log-Mel Spectrogram. We decided to use this type of representation because during the experiments it was found that the value of the index was always very close to -1 or 1 for every feature, values that indicate respectively that the feature considered has a more important impact on the other classes than the class of interest or that it is highly focused in describing the class of interest. So including the value of *nPIRP* in the graph doesn't add any useful information and makes it harder to read.

Figure 4.19 shows the *numerical explanation* of a 5s correctly classified audio of a *crying baby* (coi = 20) using *Spectrogram Features Extraction* technique (method = 3). It also showed the mean value of *nPIRP*. The x-axis shows the ID of each feature extracted for each different patch. The indexes are not incremental on the whole spectrogram but start again from index 1 for each patch for reasons of readability of the graph.

The y-axis shows the value of the *nPIR* index for the respective feature.

On top of the graph several informations are provided :

- *Class of Interest* : the corresponding index of the class of interest on which the classification analysis is carried out.
- *Method* : indicates the feature extraction technique used in the analysis.
- *nPIRP mean* : show the mean value of the *nPIRP* index calculated over all the features extracted on the spectrogram.

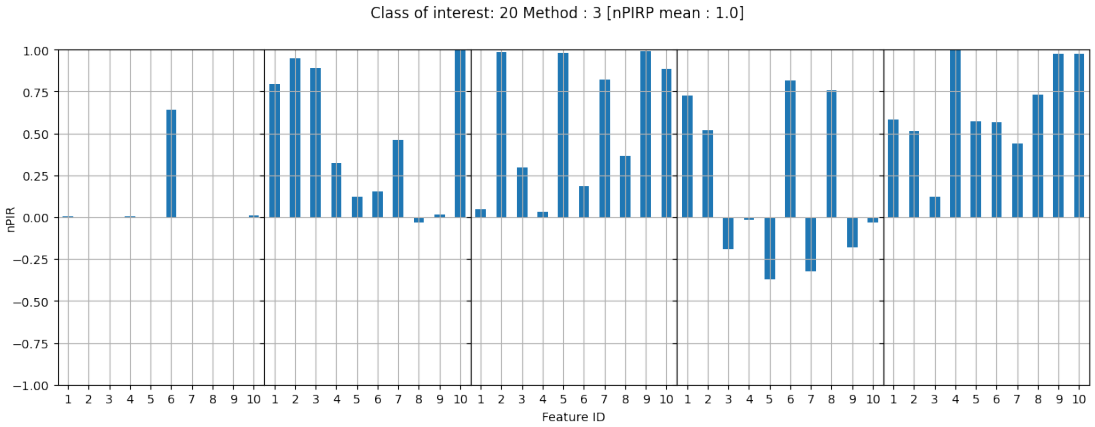


Figure 4.19: Numerical Explanation example

Chapter 5

Experimental Results

In this chapter are discussed the experimental results observed after manually inspecting 200 audio taken from ESC50 Dataset and 50 audio from UrbanSound8K Dataset.
In the first section we describe the experimental settings, that we show 3 examples deeply analyzed. Other examples show peculiar A-EBAnO explanations. We then describe a comparison between two models trained on the two different Dataset and finally an automated global analysis of the quality of the features extracted by A-EBAnO.

5.1 Experimental Settings

In this section are explained the implementation choices to produce local explanation in context of *audio classification* using *A-EBAnO*, explained in section 4.2.

Classifiers During the experiments we produced local explanation using 2 models. The classifier architecture is explained in section 3.3. The classifier is build upon *Vggish*, a slightly modified version of VGG, a popular Neural Network used in image classification. The 2 models differ in the Dataset used during the training.

Datasets and Task The two Dataset used in the experiments are *ESC50*, described in subsection 3.4.1 and *UrbanSound8K*, explained in subsection 3.4.2. They are the same Dataset used during the training of the classifier.
Both Datasets are composed of audio clips of *environmental sounds* so the classification *task* for which the local explanations are produced is *Environmental Sound recognition*, explained in section 3.1.

Number of layers and extracted features Model used in the experiments have a total of 6 *convolutional layers*. These layers contains the knowledge that we want to mine to build up the local explanation of the classification. Using all the layers can produce a very deep *tensor* that is difficult to manage. Moreover the most characterizing informations of the model are stored in the deepest layers of the network. This led our analysis to be focused on the *3 deepest convolutional layers* of the classifier.
A fair upper limit of the number of *extracted features* is necessary in order to be able to build an explanation composed by meaningful semantic features. A too large or small number of features leads to an explanation with an unclear meaning for the user. During the experiments we set a maximum number of interpretable features to 10. We think is a fair trade-off between cost of computation and interpretability of the explanations. All explanations will be formed from features in a range from 2 to 10 and the most informative explanation for every patch of the Log-Mel spectrogram is chosen by A-EBAnO and combined with the most informative explanation of all other patches forming the final explanation of the entire spectrogram.

5.2 Explanation Examples

In this section are deeply analyzed 3 different type of explanation: one right explanation, explanation with an input audio clip with presence of silence and an explanation of a wrong prediction.
For every example are presented the input Log-Mel Spectrogram, top-5 predictions based on the mean value of the predictions probabilities over all the patches forming the input, visual and numerical explanations.
Other explanations showing peculiar performance of A-EBAnO are then presented.

5.2.1 Example of right prediction

This example show the explanations of a correct prediction on the class *Crying Baby*, with class ID equal to 20. Log-Mel spectrogram is show in Figure 5.1 and top-5 prediction probabilities are shown in Table 5.1

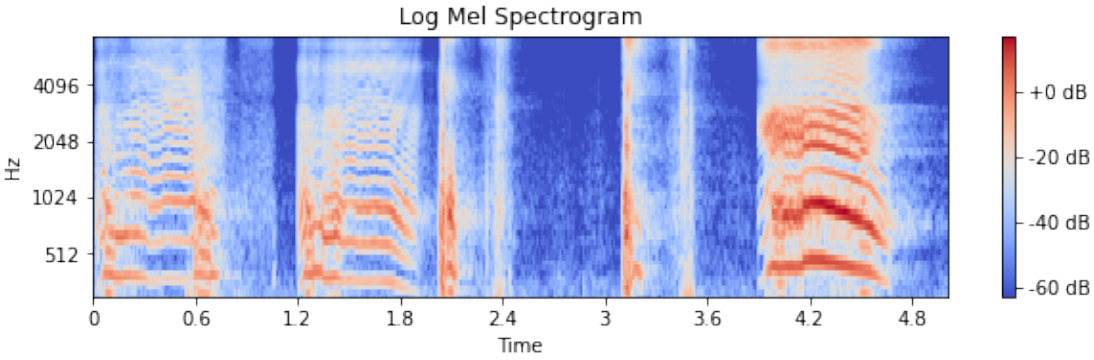


Figure 5.1: Log-Mel Spectrogram Crying Baby

ORIGINAL LABEL	PREDICTED LABEL	PROBABILITY
Crying Baby	Crying Baby	0.95
	Door wood creaks	0.00
	Dog	0.00
	Rooster	0.00
	Pig	0.00

Table 5.1: Prediction probabilities Crying Baby (COI = 20).

Single Patch Features Extraction Explanations In this section are showed visual and numerical explanations using *Single Patch Features Extraction* technique (Method = 0). Figure 5.2 and Figure 5.3 compose Visual explanation while Figure 5.4 shows the Numerical Explanation.

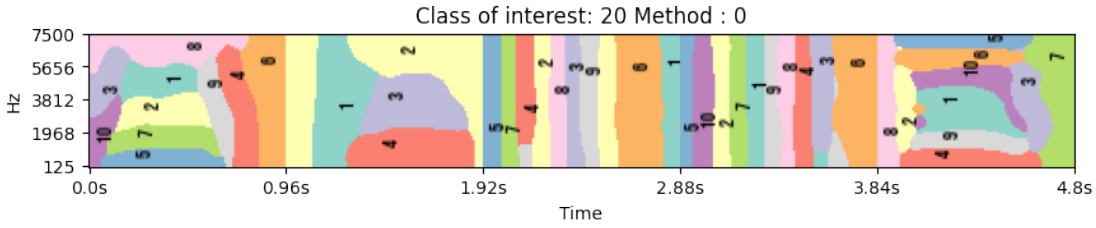


Figure 5.2: Interpretable Features Map Crying baby using method 0

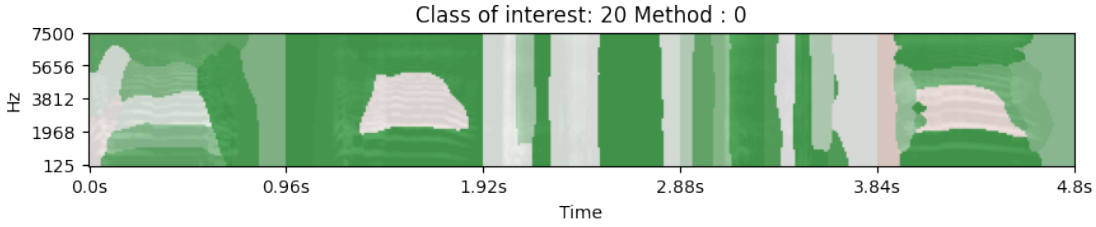


Figure 5.3: Visual Explanation Crying baby using method 0

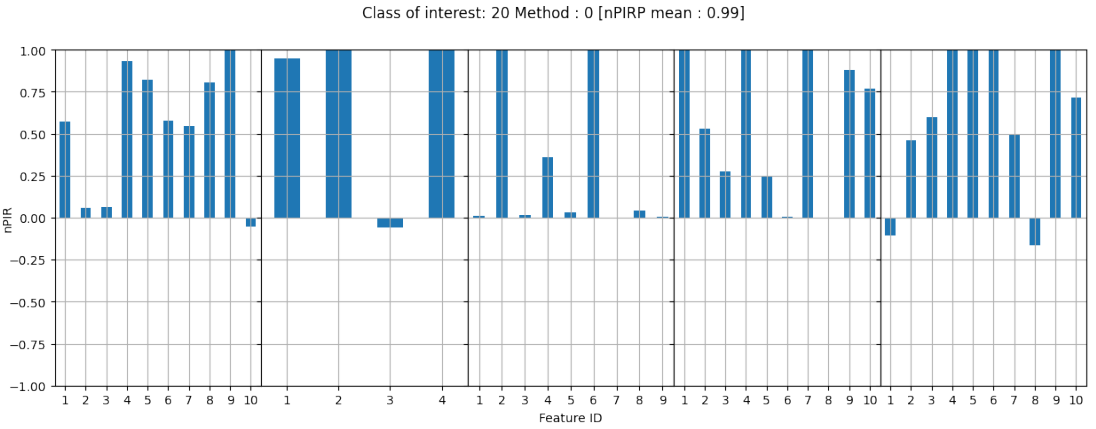


Figure 5.4: Numerical Explanation Crying baby using method 0

Analyzing the explanations we can see that A-EBAno is able to show to us that the most important parts of the clip for the classification are the first two seconds

and the last second. Comparing features map to numerical explanation we can see that nPIR value on this portions of the clip are very close to 1 and represented with a vibrant green. In this two segments of the clip is clearly audible a sound of a *crying baby*. Interesting to note is that from around second 2 to the end of second 3 A-EBAnO correctly show portions of the spectrogram that have a neutral or negative influence on the prediction indeed in this section the sound is very different because the baby stop for a moment to cry and emits two coughs. The last patch representing last second of the clip is also described with a slightly negative influence features with ID equal to 1 and 8. In this last part the baby makes a shrill sound that the model classify better as a door creaks. The interpretable features extracted by A-EBAnO are very focused on describing label *Crying Baby* with a mean *nPIRP* value of 0.99.
With this type of technique we can understand also that in general A-EBAno produces explainable features that represent both frequencies bands and time bands . In this example the most informative portions of the spectrogram are represented mostly with features that develop horizontally on the frequency axes while in the portions of the clip where there are portions of sound that do not belong to the class of audio input, A-EBAnO extract features that develop on the time axes.

Frequency Bands Features Extraction Explanations Using this features extraction technique (method = 1) we can better focus on the analysis of the frequencies covered by the sound in the clip on the Log-Mel scale.

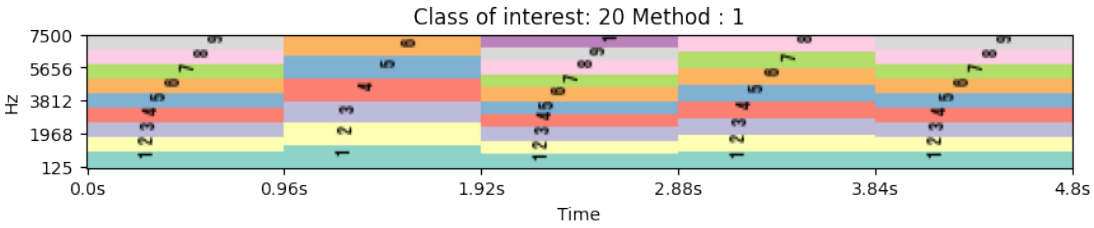


Figure 5.5: Interpretable Features Map Crying baby using method 1

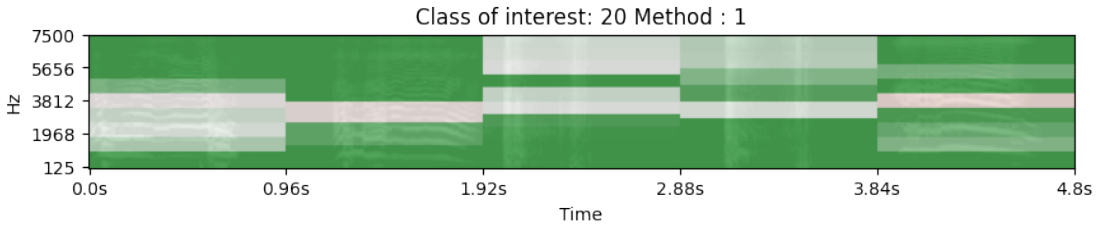


Figure 5.6: Visual Explanation Crying baby using method 1

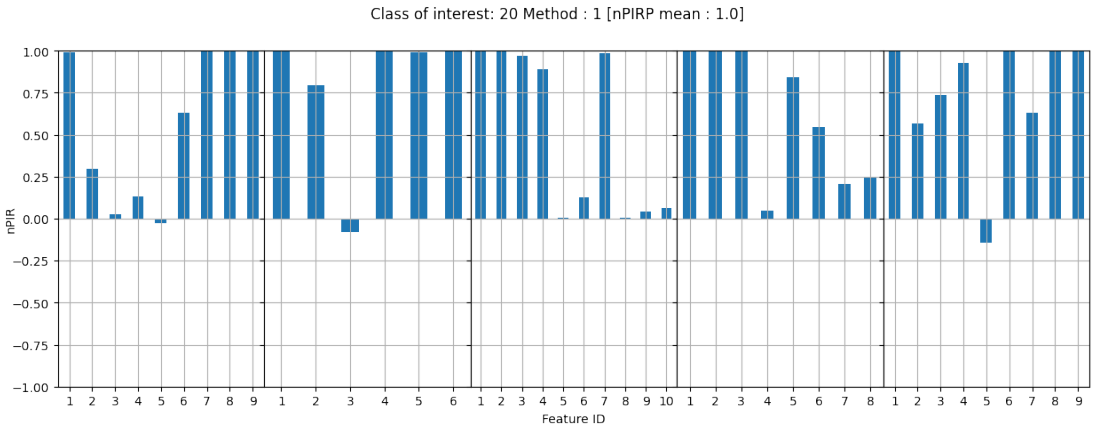


Figure 5.7: Numerical Explanation Crying baby using method 1

We can see with the visual explanation, represented in Figure 5.5 and Figure 5.6, A-EBAnO dynamic features band division over the patches composing the spectrogram. Every patch it is divided in the most informative number of frequency bands. Explanations produced using this features extraction techniques show to use what frequencies bands are the most important in the classification. Over all the clips low frequencies are always important in classification, typical of the sound of a crying baby. In the first two and the last second also high frequencies have a positive influence with *nPIR* values represented in the numerical explanation in Figure 5.7 with a value of 1. In the central part of the clip high frequencies have a neutral influence on the class of interest classification because during this moment of time the baby do not cry but emits cough and inhale air. In the last second we can see that mid frequencies have a negative impact while on the whole clip mid frequencies have a neutral influence. The precision of extracted features is very high, with a mean *nPIRP* value of 1.0.

Time Bands Features Extraction Explanations Explanations produced using this features extraction method (method =2) better highlights the instants of time that influence the model in the input classification.

Analyzing explanations produced with this features extraction technique we can see in the visual explanations in Figure 5.9 based on the features map in Figure 5.8 that the classification result is mostly positively influenced by the last moment instant of the baby's first wailing and the first moment of time of the baby's second wailing. At the end of the second wailing the child emits the two coughs and this instants of time are correctly estimated neutral for the classification. At the start of the last second we have a negative influence time bands while the rest of the clip have a positive influence in the classification as we can see in the numerical explanation in Figure 5.10, where *nPIR* values of the central extracted features on the patch, with ID range from 2 to 7 , are all 1 or close to 1.

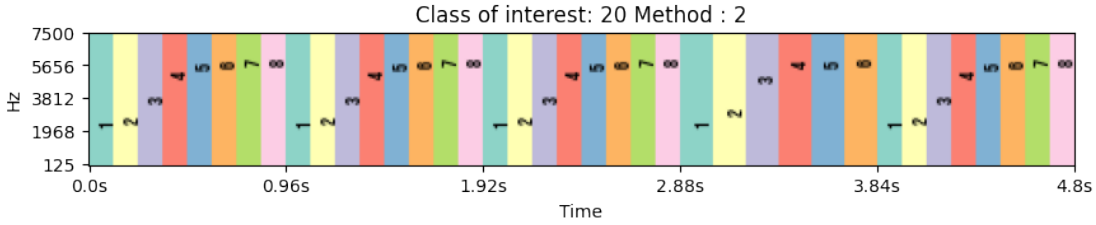


Figure 5.8: Interpretable Features Map Crying baby using method 2

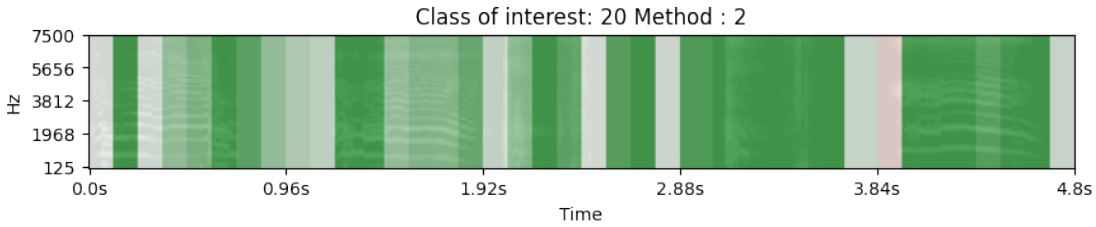


Figure 5.9: Visual Explanation Crying baby using method 2

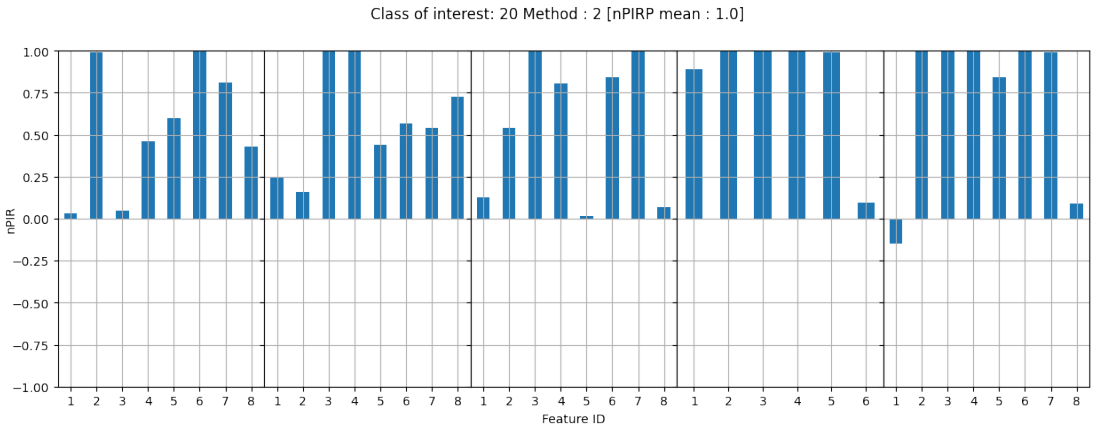


Figure 5.10: Numerical Explanation Crying baby using method 2

Spectrogram Features Extraction Explanations Unlike the first way of features extraction, using this type of technique (method = 3) the extracted features are calculated by taking as input the hypercolumns of the complete spectrogram of the input audio.

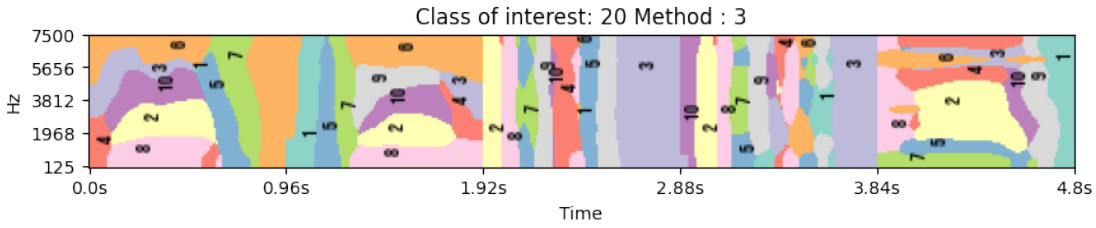


Figure 5.11: Interpretable Features Map Crying baby using method 3

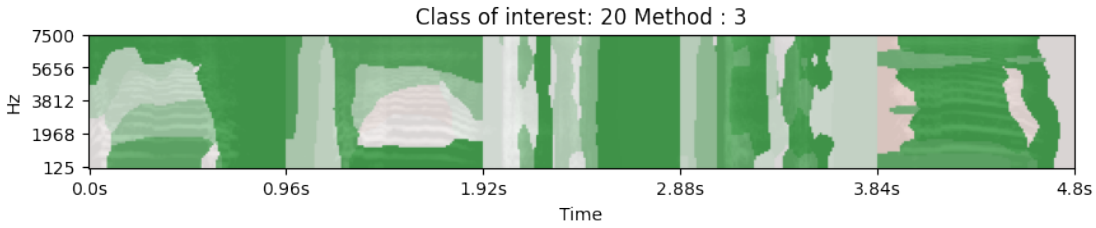


Figure 5.12: Visual Explanation Crying baby using method 3

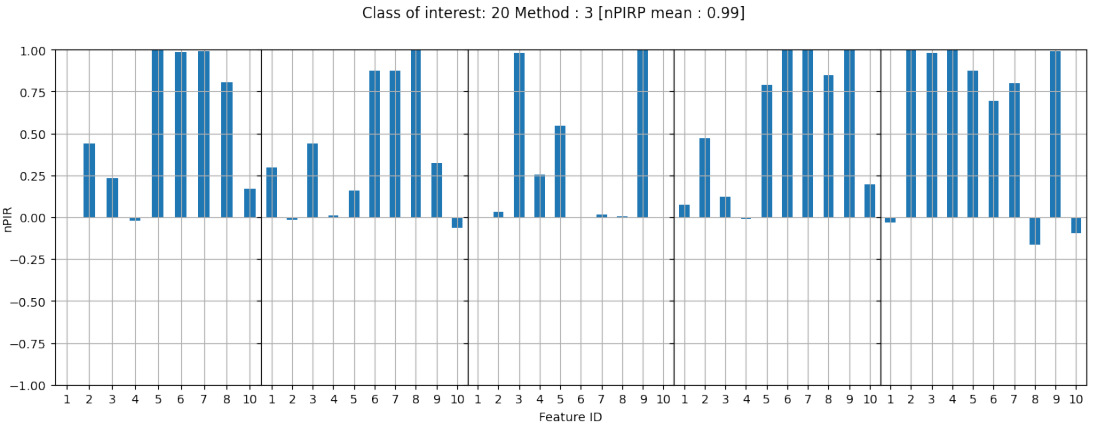


Figure 5.13: Numerical Explanation Crying baby using method 3

We can see that similar to the explanations produced using method 0, the interpretable features present a development on the frequency axis for the parts of

the clip that have a very positive influence on the classification, i.e. the first and last seconds. In the middle part, where the sound type is different from the audio class, the features have a development on the time axis.
The biggest difference is in the latest patch. In this case all the part of the spectrogram representing the vagus of the newborn correctly presents a positive influence, while the method 0 presented it negatively influential. The start and the end of the patch are the portion of the spectrogram that have a negative influence, corresponding to the same portions recognized using method 2. Mean value of *nPIRP* is also in this case very high with value equal to 0.99

Comparing it with method 0 , we can say that *Spectrogram features extraction* method produces interpretable features that describe the spectrogram in more detail and in a more precise way and therefore we consider it preferable to method 0.

5.2.2 Example of audio clip with silence

This example show how A-EBAnO is able to understand and analyze portions of the input audio spectrogram in which audio is actually present, completely ignoring the silent parts of the clip. Log-Mel spectrogram and top-5 prediction probabilities are shown in Figure 5.14 and Table 5.2 and visual and numerical explanations using *Time Bands Features Extraction* (method = 2) and *Spectrogram Features Extraction* (method = 3) techniques of an audio clip belonging to class *Coughing*, with class ID equal to 24 are provided.

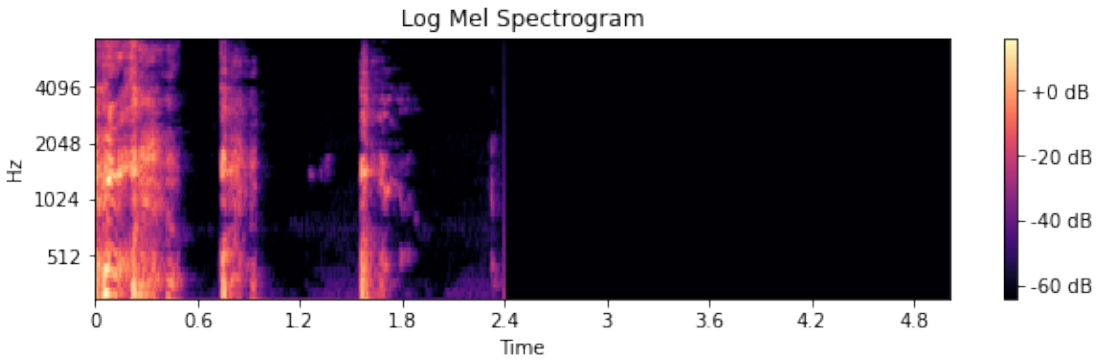
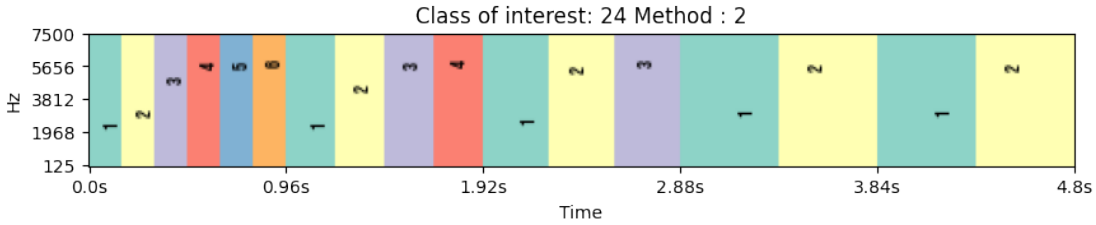


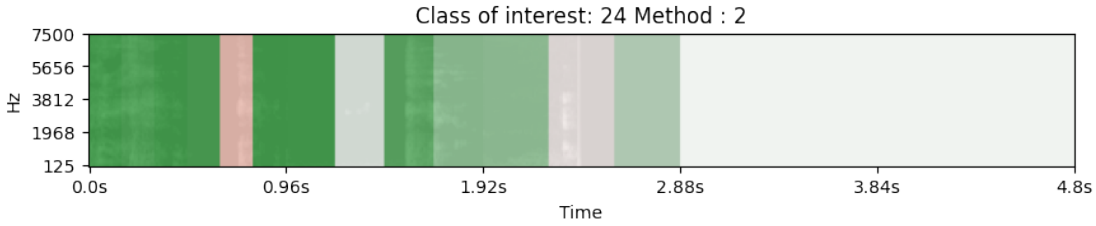
Figure 5.14: Log-Mel Spectrogram Coughing

ORIGINAL LABEL	PREDICTED LABEL	PROBABILITY
Coughing	Coughing	0.80
	Glass breaking	0.34
	Sneezing	0.31
	Door wood knock	0.29
	Rooster	0.28

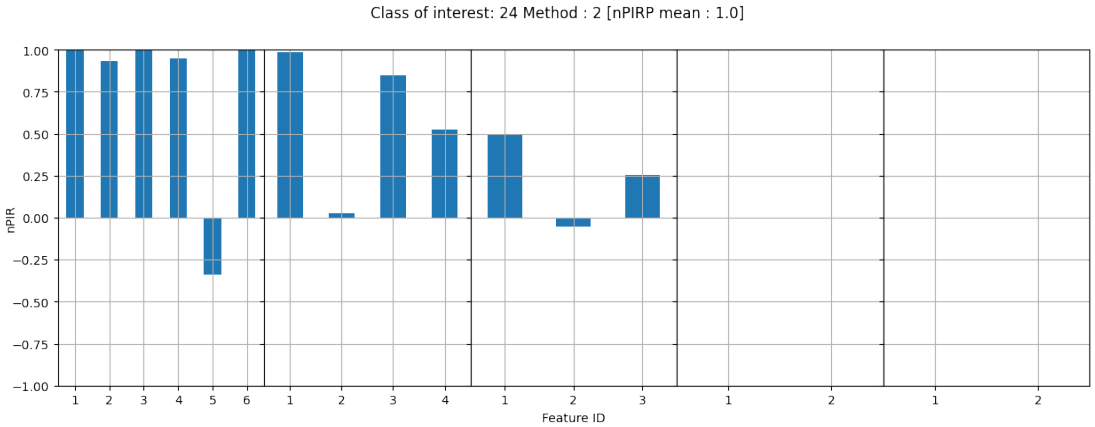
Table 5.2: Prediction probabilities Coughing (COI = 24)



(a) Features Map Coughing

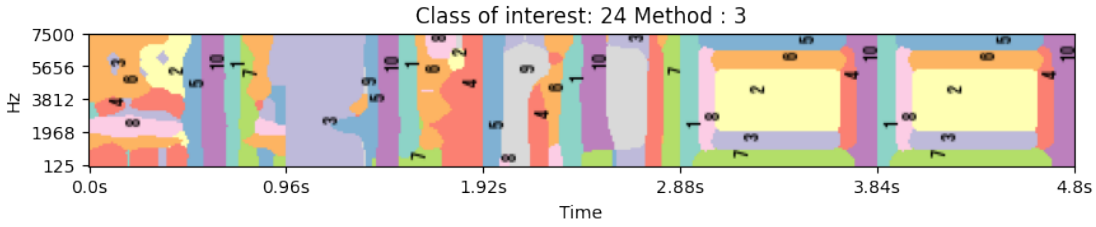


(b) Visual Explanation Coughing

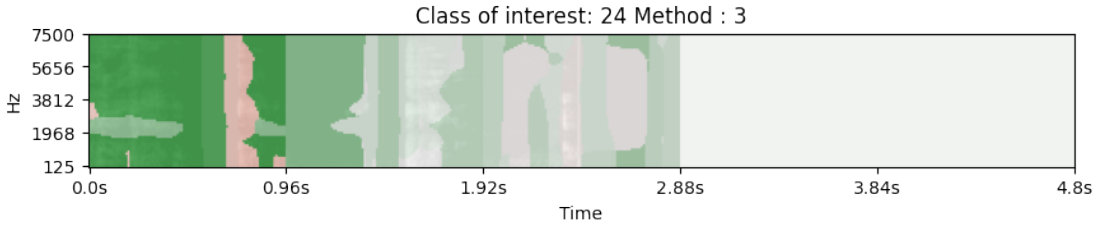


(c) Numerical Explanation Coughing

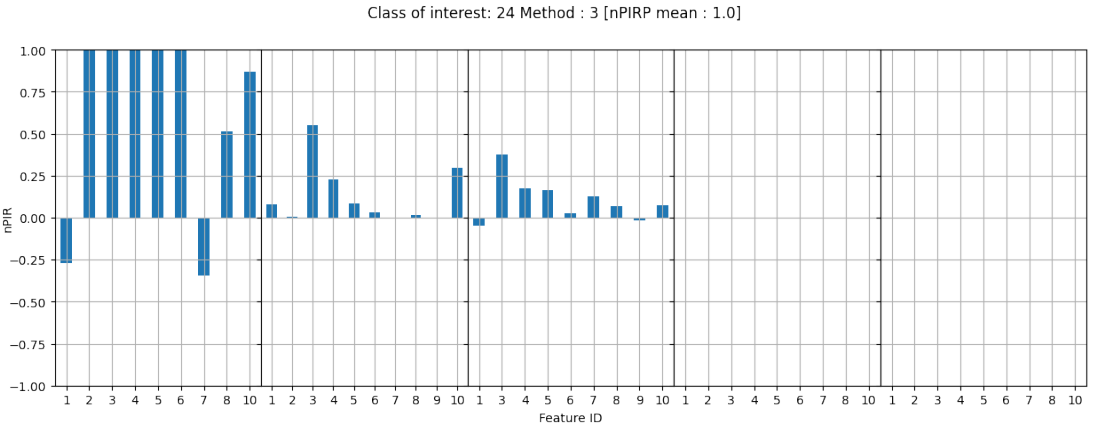
Figure 5.15: Visual and Numerical Explanations Coughing using method 2



(a) Features Map Coughing



(b) Visual Explanation Coughing



(c) Numerical Explanation Coughing

Figure 5.16: Visual and Numerical Explanations Coughing using method 3

Analyzing explanations produced using both interpretable features extraction techniques we can see that A-EBAnO is able to distinguish portions of the Log-Mel spectrogram that actually contain information to be analyzed from those that do not have any sound.
Last two seconds of the clip are silent and A-EBAnO correctly explain to us that that part of the audio input is completely ignored. The first patch positively influence the result of the prediction with values of *nPIR* equal to 1 to the most features extracted on this clip by both techniques.
Both analyses show that after this initial cough there is a moment of time that

negatively affects classification. In this moment in time, a type of guttural sound is distinguishable that differs from the typical coughing sound. In method 2 is represented with the feature ID number while using method 3 it is represented with an ID equal to 7. In the clip there is then a moment of silence that is recognized as neutral, with ID equal to 2 in Figure 6.15a and then followed by a final quick cough that again is recognized as positively influencing the classification result.

Important to notice is that in both analysis after the last cough a small portion of the spectrogram that is silent has a slightly positive influence . Analysing other audio input has emerged that types of sound that can be characterized as *impulsive*, like coughs, have created a cognitive *bias* during the training of the model. In fact, the model tends to recognize silence just after all types of impulsive sounds as positively influential in classification. WE can also recognize this bias by looking at the classification probability table. The top-3 predicted label all belong to 'impulsive' type of sound.

5.2.3 Example of wrong prediction

This is an example of incorrect classification explanation. The original label of the audio input is *Cat*, but the classifier incorrectly classify the input as *Glass breaking*. In Figure 5.17 is reported the Log-Mel Spectrogram and top-5 prediction are showed in Table 5.3. Explanations using *Frequency Bands Features Extraction* (method = 1) and *Spectrogram Features Extraction* (method = 3) are reported after prediction probabilities table. The first analysis is performed using the wrong label 'glass breaking' with label ID equal to 39 as the class of interest. Then the analysis is shown using the original 'cat' label as class of interest with label ID equal to 5.

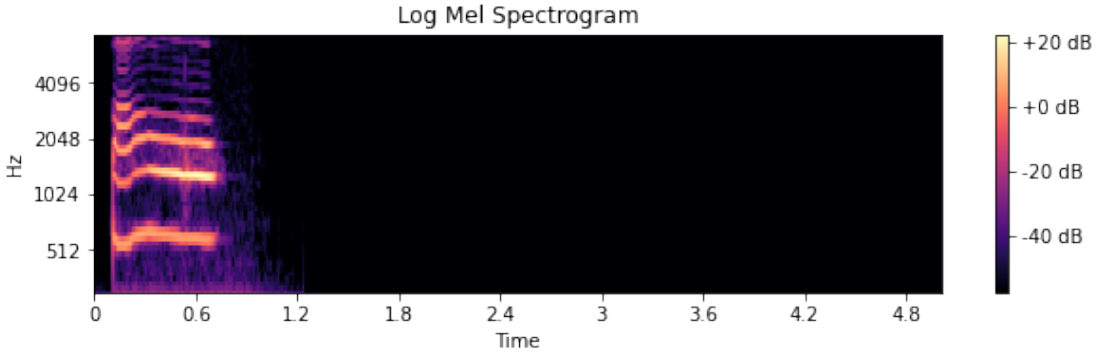
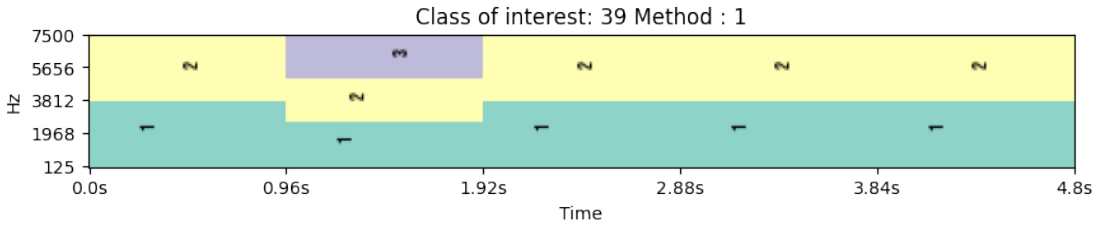


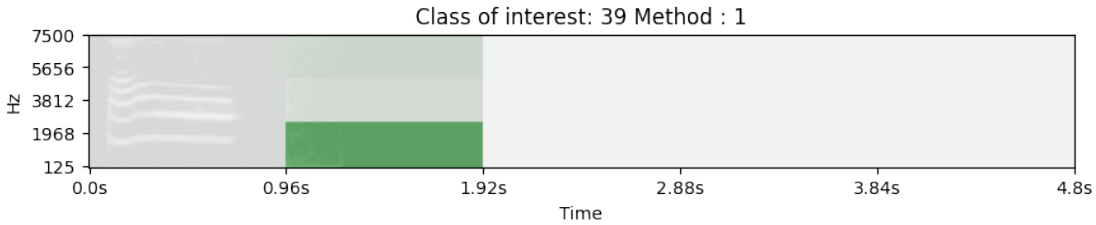
Figure 5.17: Log-Mel Spectrogram Cat

ORIGINAL LABEL	PREDICTED LABEL	PROBABILITY
Cat	Glass breaking	0.52
	Door wood knock	0.50
	Sneezing	0.47
	Cat	0.44
	Coughing	0.43

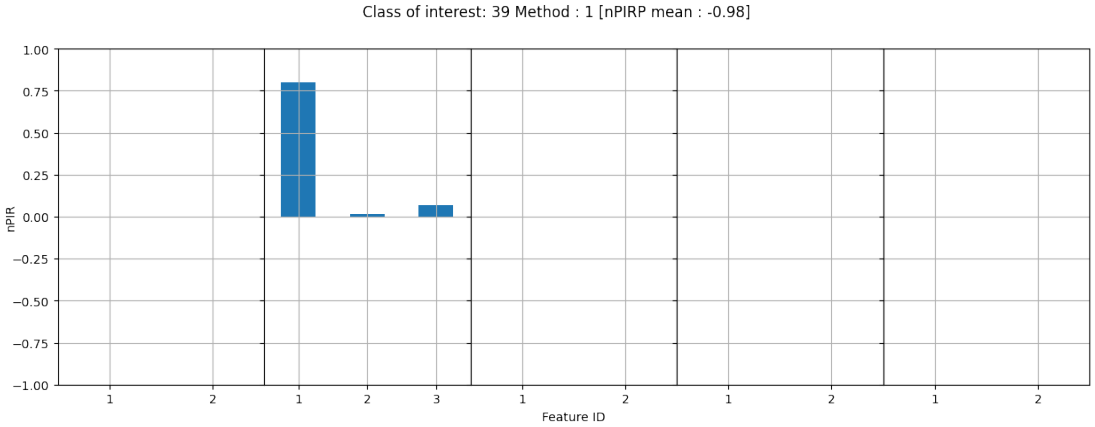
Table 5.3: Prediction probabilities Cat (COI = 5). Glass Breaking COI = 39.



(a) Features Map coi = Glass Breaking

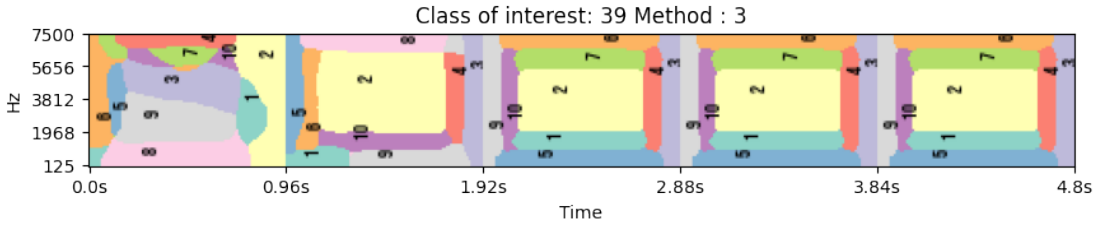


(b) Visual Explanation coi = Glass Breaking

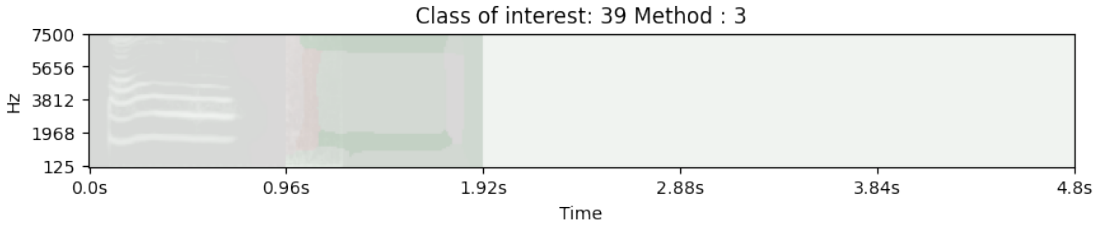


(c) Numerical Explanation coi = Glass Breaking

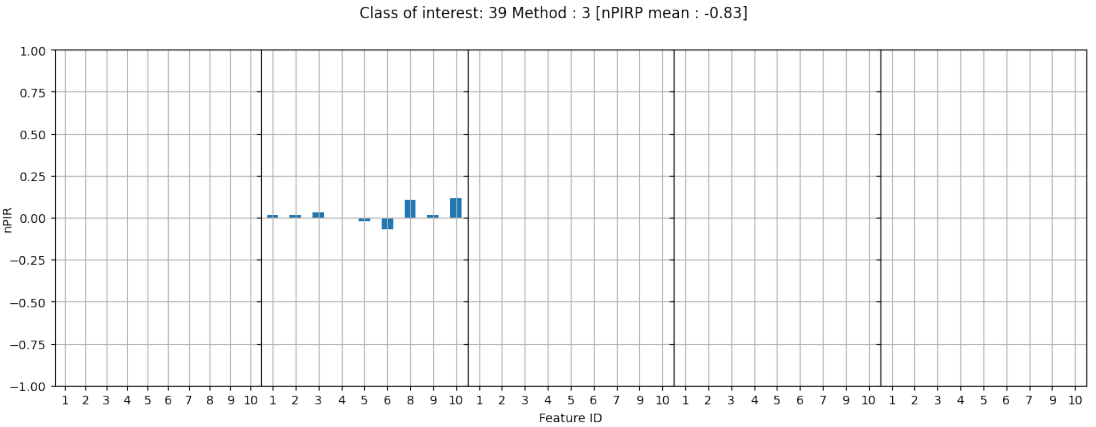
Figure 5.18: Visual and Numerical Explanations using method 1 with coi = 'Glass Breaking'



(a) Features Map coi = Glass Breaking



(b) Visual Explanation coi = Glass Breaking



(c) Numerical Explanation coi = Glass Breaking

Figure 5.19: Visual and Numerical Explanations using method 3 with coi = 'Glass Breaking'

Analyzing firstly the explanations produced using as class of interest the wrong predicted label 'Glass Breaking' we can understand that the the error is due to the first moments of silence following the cat's meow. Both features extraction techniques shows that the features that affect classification the most are those represented by the low and high frequency bands in the second patch, which starts just after the end of the meow. Explanations produced using *Frequency Bands Features Extraction* (Figure 5.18) highlight how much influence is feature with id equal to 1 that reach a *nPIR* value over 0.75. *nPIR* index values reached by

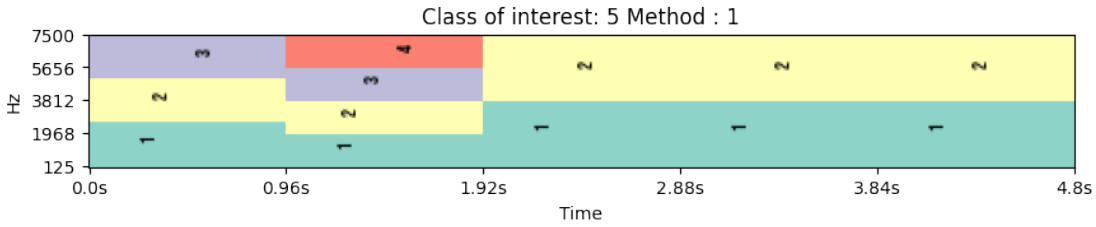
features number 8 and 10 in the numerical explanation using *Spectrogram Features Extraction* (Figure 5.19) technique are lower but still positive. Their effect is mitigated by the fact that the analysis is performed on the full spectrogram, where their positive influence have a minor impact on the classification.

Is important to highlights that the mean of *nPIRP* value on both the analysis is very negative : -0.98 using *Frequency Bands Features Extraction* method and -0.83 using *Spectrogram Features Extraction* method. This mean that the classifier is very is very uncertain about the ability of the extracted features to describe significant portions of the input.
These explanations give further evidence to the claim that the model is biased when it has to classify a type of audio that has a very short sound followed by a long silence.

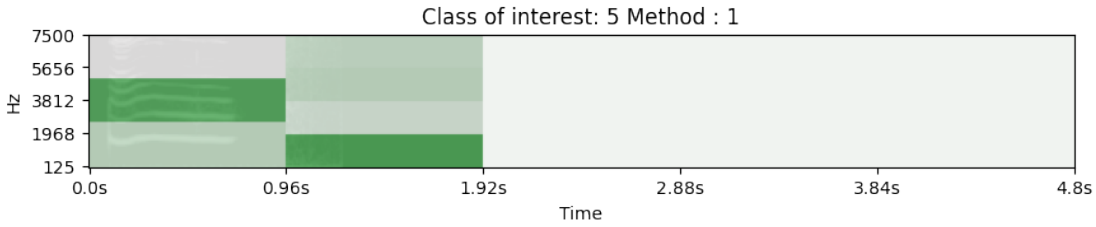
Now we analyze the explanations produced using 'Cat' as class of interest in order to understand why the model is unable to classify the sound correctly. We show explanations using again *Frequency Bands Features Extraction* (Figure 5.20) and *Spectrogram Features Extraction* (Figure 5.21) techniques to have a fair comparison.

Studying the explanations using as class of interest the label 'Cat' we notice that now portion of the spectrogram that correctly represent the cat's meow are recognized as positively influential. Looking at the explanation produced using *Frequency Bands Features Extraction* technique two over three extracted features in the first patch have positive values of *nPIR*. Especially features with ID equal to 2 have a *nPIR* very close to 1. The same portion of the spectrogram is recognised also using *Spectrogram Features Extraction* method has the same positive value but slightly less than 0.5.

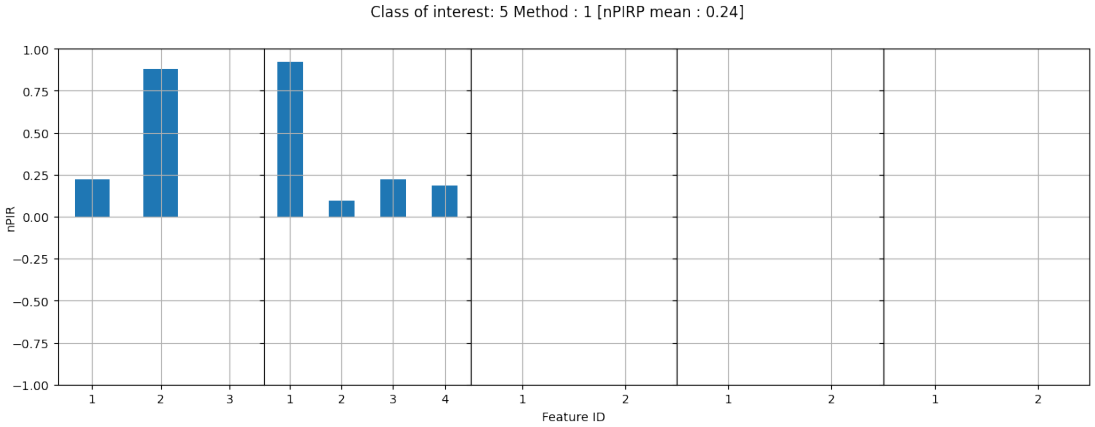
The important reason because the model can't correctly recognise the sound in the audio is that the mean of *nPIRP* is low for both the explanations : 0.24 using method 1 and 0.26 using method 3.
When the model have not a mean value of *nPIRP*, so it is not completely certain that the extracted features are focused on describing only the class of interest, it prefers to classify the sound in the patch as 'Glass breaking', because of the bias acquired during training.



(a) Features Map coi = Cat

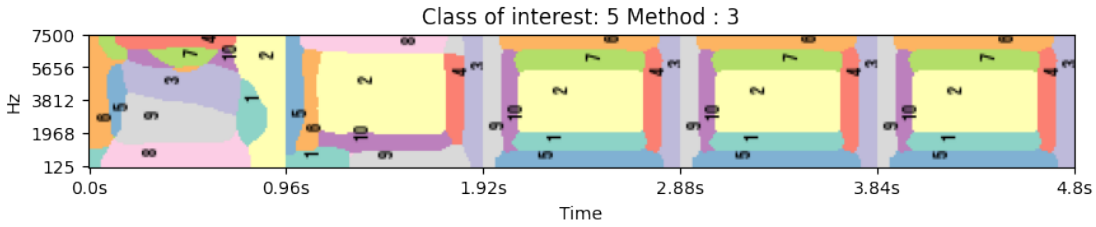


(b) Visual Explanation coi = Cat

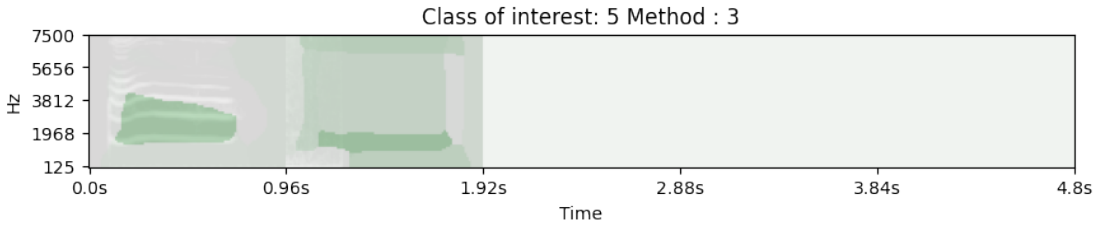


(c) Numerical Explanation coi = Cat

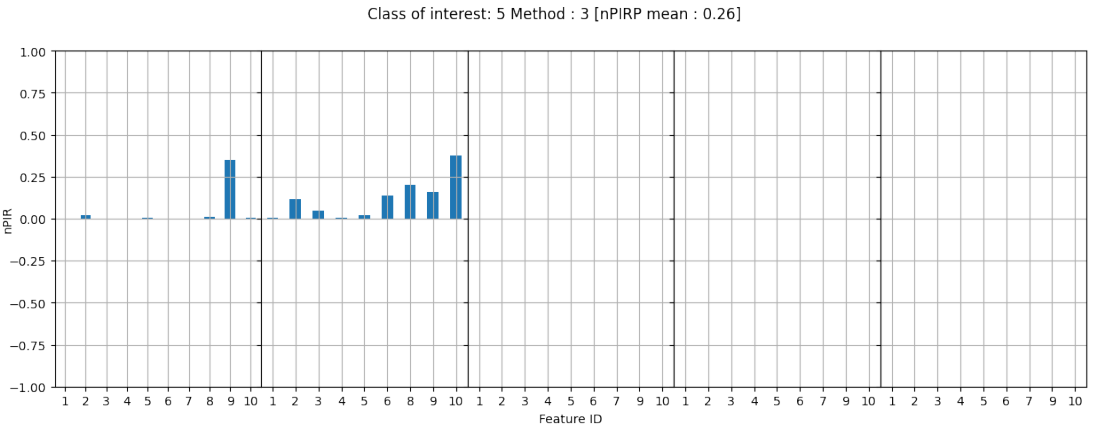
Figure 5.20: Visual and Numerical Explanations using method 1 with coi = 'Cat'



(a) Features Map coi = Cat



(b) Visual Explanation coi = Cat



(c) Numerical Explanation coi = Cat

Figure 5.21: Visual and Numerical Explanations using method 3 with coi = 'Cat'

5.2.4 Other Relevant Examples

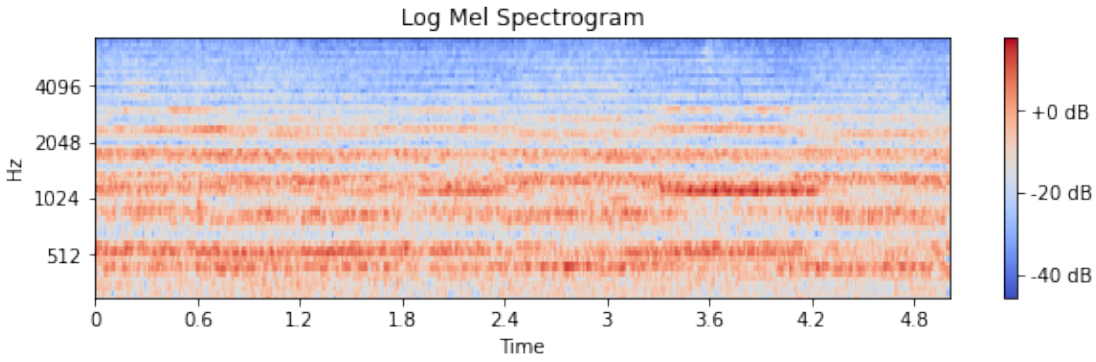
In this section we show 3 other examples in which we can see the ability of A-EBAnO to find features that can describe in a clear and precise way the input audio and to build in this way an explanation useful to the user in understanding the result of the classification. For each examples the spectrogram, top-3 prediction and visual explanation are provided using the most suitable features extraction technique for the explanation.

Siren (frequencies) In this example we can see how A-EBAnO explanations shows that the correct classification of input audio as 'Siren' sound is due to the interpretable features that represents high frequencies as we can expect listening to the sound of a siren, characterized by high frequencies sounds. Table 5.4 show the classification probabilities for the top-3 predicted label.

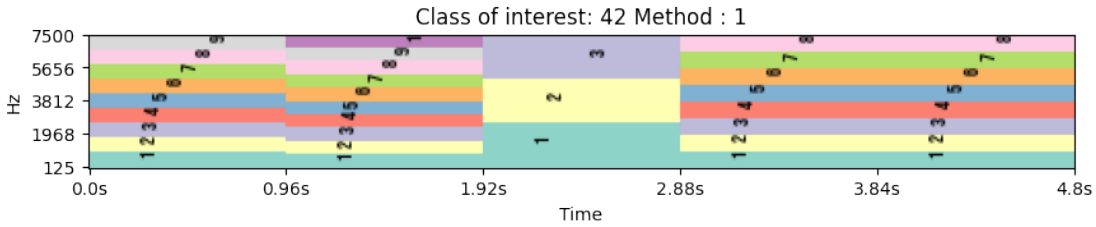
Figure 5.22(c) shows the visual explanation of the classification. On over the clip high frequencies have a positive influence on the classification with nPIR values near to 1. In the third patch is present a portion of the low frequencies represented in the spectrogram that also have a positive influence. It is due to the presence of second 'Siren' sound that that comes from a source further away from the place where the clip was recorded , which could be described by lower frequencies.

ORIGINAL LABEL	PREDICTED LABEL	PROBABILITY
Siren	Siren	0.99
	Car horn	0.00
	Church bells	0.00

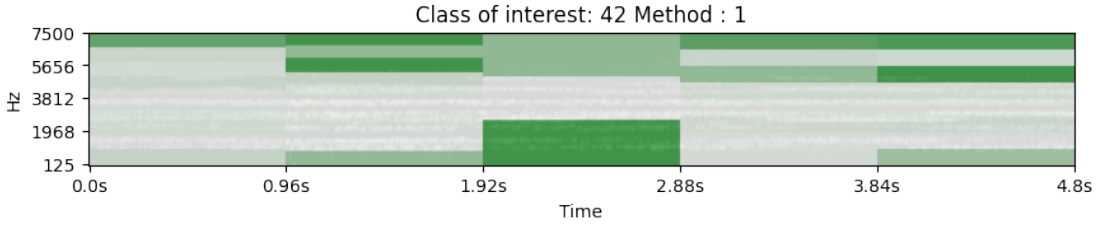
Table 5.4: Prediction probabilities Siren (frequencies)(COI = 42).



(a) Log-Mel Spectrogram Siren (frequencies)



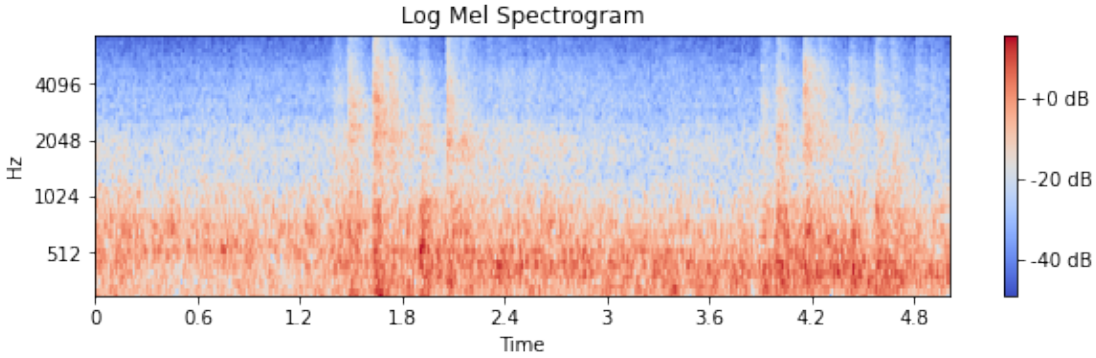
(b) Features Map Siren (frequencies)



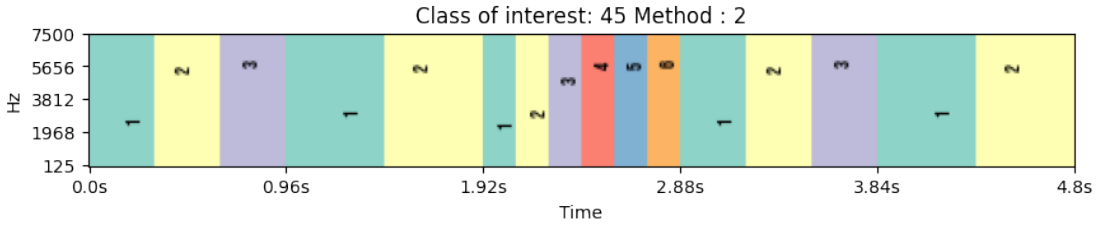
(c) Visual Explanation Siren (frequencies)

Figure 5.22: Log-Mel Spectrogram and Visual Explanation Siren using *Frequency Bands Features Extraction* technique

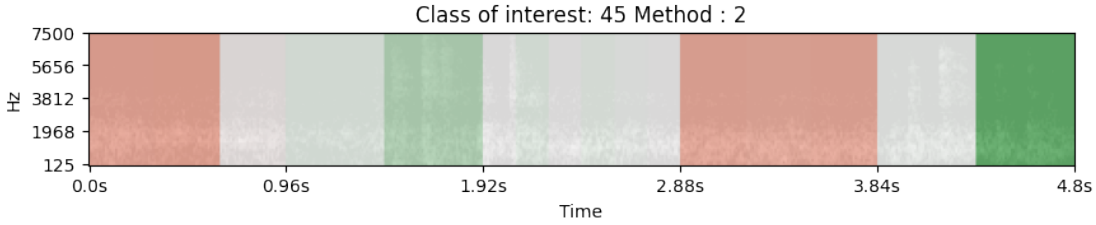
Train This example emphasizes the use of *Time Bands Features Extraction* technique to clearly understand what moments of time during the audio clip are the most important for the classification. The input audio belong to the class 'Train'. Top-3 predicted class are reported in Table 5.5.



(a) Log-Mel Spectrogram Train



(b) Features Map Train



(c) Visual Explanation Train

Figure 5.23: Log-Mel Spectrogram and Visual Explanation Train using *Time Bands Features Extraction* technique

ORIGINAL LABEL	PREDICTED LABEL	PROBABILITY
Train	Train	0.82
	Wind	0.11
	Dog	0.00

Table 5.5: Prediction probabilities Train (COI = 45)

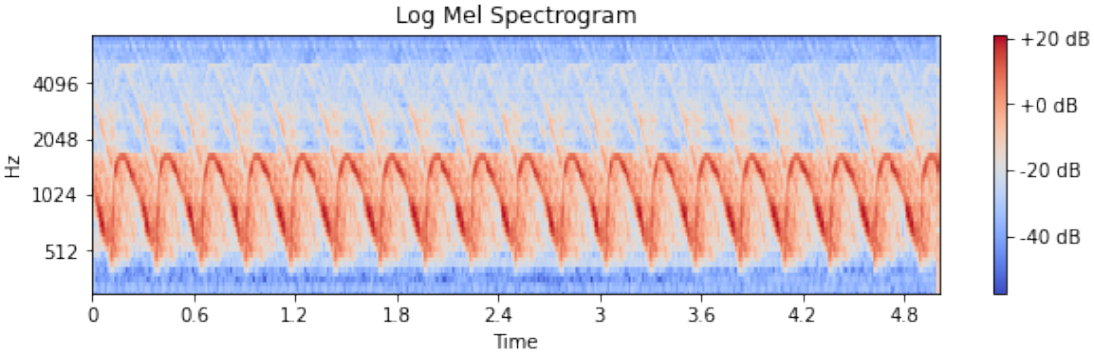
Analyzing the visual explanation in Figure 6.23(c) we understand that the model is focused on two precise moment of time , around second 2 and at the end of the clip. In this moments of time is clearly audible the sound of the train while crossing a junction between tracks. It is this type of sound present in the audio clip that the classifier utilize to correctly classify the audio as 'Train'. Features representing time bands with negative influence represent represent instants of time during which the sound of the running train can be mistaken for that of the wind.

Siren (Spectrogram segmentation) This final example want to show how A-EBAnO *Spectrogram Features Extraction* method can find interpretable features that can precisely follow and describe the distribution of the sound power on the spectrogram. The input sound belong to the class 'Siren' and top-3 predicted classes are reported in Table 5.6.

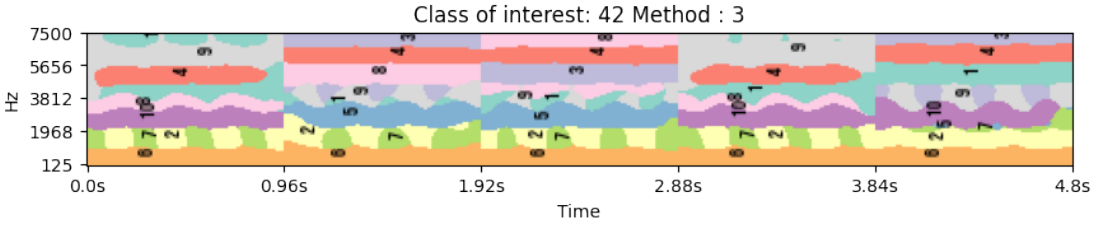
ORIGINAL LABEL	PREDICTED LABEL	PROBABILITY
Siren	Siren	0.99
	Dog	0.00
	Rooster	0.00

Table 5.6: Prediction probabilities Siren (COI = 42)

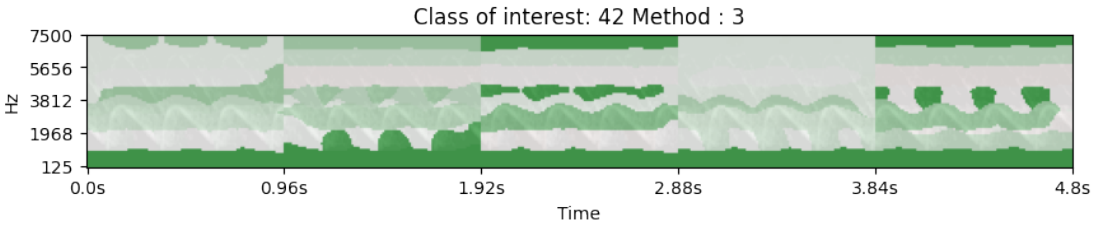
Looking at the Log-Mel spectrogram in Figure 6.24(a) we can easily understand that could be difficult to find interpretable features that can correctly describe the power distribution of the sound in the spectrogram. However the features map that we can see in Figure 6.24(b) shows that A-EBAnO its able to find features that highlights correctly the portion of the spectrogram that represent the power distribution.



(a) Log-Mel Spectrogram Siren



(b) Features Map Siren



(c) Visual Explanation Siren

Figure 5.24: Log-Mel Spectrogram and Visual Explanation Siren using *Spectrogram Features Extraction* technique

5.3 Comparison between models

In this section we show a comparison between the explanations of classification result on a input audio, taken from *ESC50* Dataset, representing a 'Dog' sound, using two instances of our classifier. The first model is trained on *ESC50* Dataset (subsection 3.4.1) and the other one is trained on *UrbanSound8K* Dataset (subsection 3.4.2).
We decide to use the model trained on UrbanSound8K that uses the same strategy of training of the model trained on ESC50 ,where the Dataset is divide in random train,validation and test sets, to have a fair comparison.

For the comparison we show the *Visual Explanation* of the classification using only the *Spectrogram Features Extraction* technique. The visual explanation contains indirectly the *nPIR* index values for every interpretable feature extracted, encoded by the color scale . Mean value of *nPIRP* value is also provided to have a complete understanding of the explanations.
This type of comparison allows us to understand the influence that can have the Dataset in the training of the model and therefore on its performances.

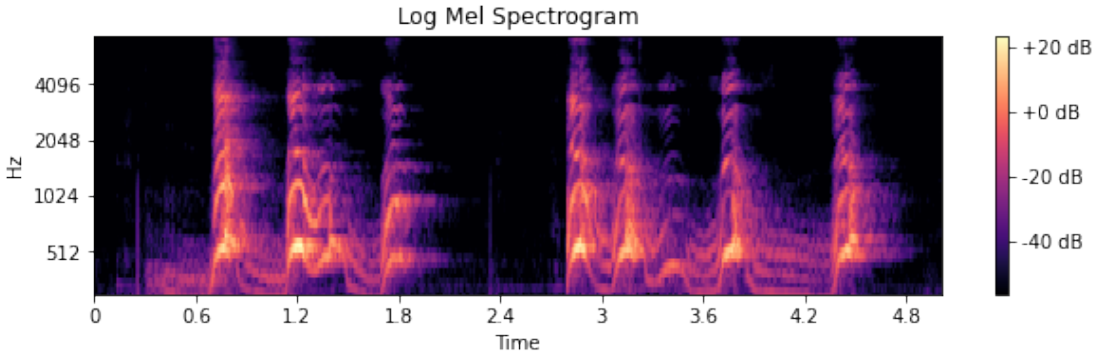


Figure 5.25: Log-Mel Spectrogram Dog from ESC50

Figure 6.25 represent the Log-Mel Spectrogram of the input audio.

The comparison baseline is the classification result and visual explanation of the classification performed by the model trained on ESC50 Dataset. Top-1 class predicted is 'Dog' with a probability of *0.99*.

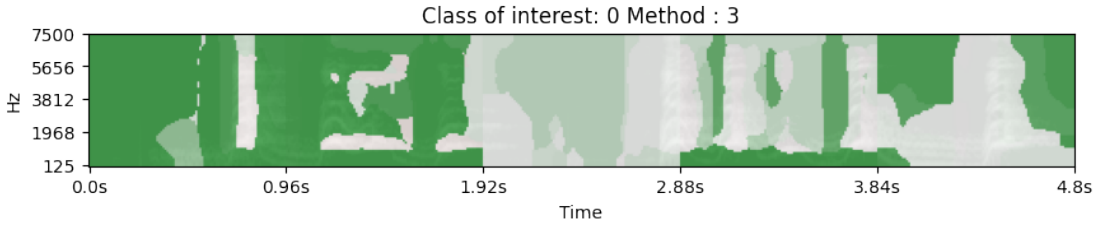


Figure 5.26: Visual Explanation of the classification performed by model trained on ESC50

The visual explanation using *Spectrogram Features Extraction* technique in Figure 5.26 show to use that the starting and ending parts of the spectrogram have a very positive influence on the classification with a neutral portion of the spectrogram in correspondence of instant of silence. Mean of *nPIRP* index is 1.0 meaning that A-EBAnO was able to find features that precisely describe the class of the audio.

Now we feed into the model trained with the other Dataset, UrbanSound8K, the same input audio
The classifier is able to correctly classify the input audio as 'Dog' with a probability equal to 0.94. The visual explanation produced by A-EBAnO is reported in Figure 5.27.

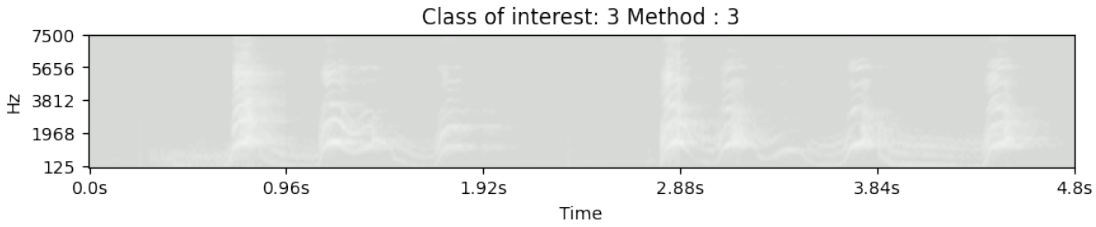


Figure 5.27: Visual Explanation of the classification performed by model trained on US8K

Looking at the visual explanation is very clear that the model absolutely don't recognize any portion of the spectrogram that can be representative of the input audio label as positively influencing the prediction. All the spectrogram have a neutral impact on the classification. This mean that any perturbed mined features change the result of the classification. Also the mean of *nPIRP* value is bad with a value equal to -1. It seems that using UrbanSound8K as a training Dataset leads to the creation of a model that is not capable of generalizing its classification, not being able to learn from precise portions of the spectrogram that positively

affect the classification result, which it may be able to identify in audio other than those used in its training. Class of interest are different because the label 'Dog' is represented by different ID's in the two Datasets used during the training

To complete the comparison we feed as input of the model trained on ESC50, a 'Dog' audio file taken from UrbanSound8K, in order to understand if the problem is the Dataset itself or if A-EBAnO its not capable to correctly perturb the interpretable features extracted from audio input taken from UrbanSound8K.

The network correctly classify the input as 'Dog' with a probability equal to *0.20*, representing the top-1 label predicted.

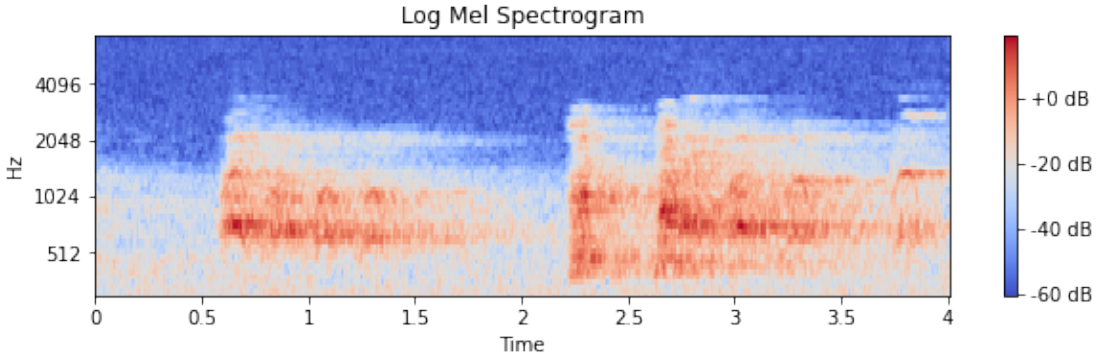


Figure 5.28: Log-Mel Spectrogram Dog from UrbanSound8k

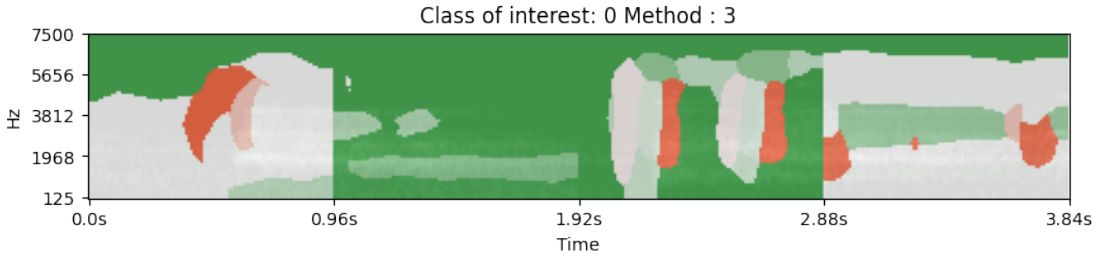


Figure 5.29: Visual Explanation Dog from UrbanSound8k performed by model trained on ESC50

In Figure 5.28 we can see the Log-Mel Spectrogram of the audio input while Figure 5.29 represent the visual explanation produced using *Spectrogram Features Extraction* method. As we can easily notice in this case A-EBAnO produce an usefull explanation of the classification. The central part of the audio clip is important in the classification while starting and ending patches are more neutral.

The mean of *nPIRP* value over the spectrogram is equal to 0.42.

Thanks to the explanations produced using A-EBAnO we can conclude that in case we should between the two models certainly our choice would fall on the model trained on ESC50, because A-EBAnO has shown us that the classifications produced are actually based on precise portions of the input and therefore our *confidence* in its results is higher.

5.4 Global Analysis

In the previous sections every example show the local explanations produced by A-EBAnO taking a single input at time.
In order to have a global vision of the proposed methodology we collect statistics over 200 audio taken from ESC50.
For each Features Extraction technique the distributions of the *nPIR* maximum and minimum values were calculated from explanations produced by A-EBAnO using as class of interest the top-1 prediction label, along with the feature distribution based on the values of *nPIR*.
Globally are analyzed 800 explanations composed of a total of 1000 different Log-Mel Spectrogram patches.

The objectives of this global analysis is to understand if A-EBAnO is able to find relevant features for the classification and to analyze the computation cost between the four different features extraction techniques looking at the computation time.

Distribution of min and max nPIR values In Figure 5.30 are showed the distributions of min and max values of nPIR computed for the all four Features Extraction techniques.
For all the methods the positively influential features, with maximum nPIR are mostly included in the [0.75,1.0] range. *Time Bands* and *Spectrogram* features extractions techniques only presents some values that fall in the [0.00,0.25] range. Rare but present are in all methods maximum values below 0.0. This means that for certain type of inputs A-EBAnO is not able to find a a positive influential feature.
Mostly of the feature with minimum nPIR are located in the [-0,25,0.25] range. Less influential features have a slightly negative or neutral impact for the classification process. Especially *Spectrogram Features Extraction* technique present near to zero minimum value of nPIR above 0.25.

The large distance between minimum and maximum nPIR values, that is responsible in the right choice of the most informative explanation presented by A-EBAnO, suggest to us that the model can produce usefull explanations.

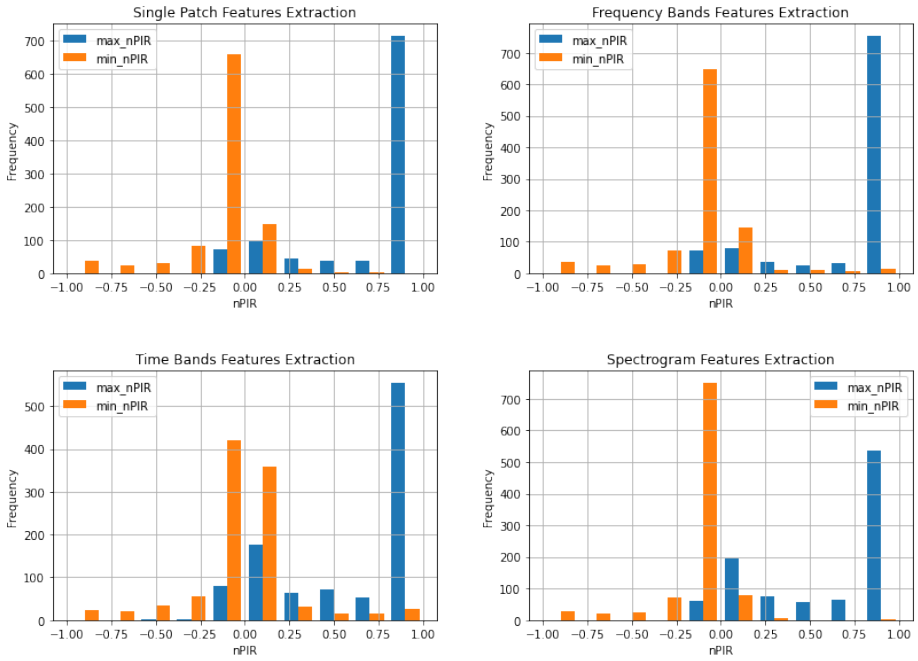


Figure 5.30: Distributions of min and max nPIR values for Features Extraction method

Distributions of nPIR values Figure 5.31 show the number of features for different bins of nPIR values, calculated using all four Features Extraction technique. All 4 distributions show that mostly of the interpretable features extracted by A-EBAnO fall in the [0.0,1.0] range. This means that the proposed methodology can mostly find features that have a neutral or positive impact on the classification result. This is not surprising as the classification metrics on this Dataset (subsection 3.4.1) show high model accuracy.

Spectrogram Features Extraction divide the input spectrogram in a larger number of features. The peak of neutral features is over 5000 for this technique while the other three shows a peak around 2000. This means that this techniques produce a more precise features map that uses to analyze the spectrogram.

In Table 5.7 are reported the percentages of features with nPIR greater or equal

to 0.50, representing positively influential features. *Frequency Bands Features Extraction* method have the highest percentage. The input of the classification are spectrogram. This type of visual representation of audio data shows the power distribution of audio power over time for different *frequencies* in the Mel Scale. Spectrogram representation is so focused on the frequencies and this result show that a division of the input in frequency bands leads to find a larger number of positive features.
The lowest percentage is reached by *Spectrogram Features Extraction* method. This is due to the number of neutral features that is more that double respect to others methods. The reason is that since this method divides the input into a larger number of features it is more likely to find both neutral and negative features, thus lowering the percentage of positive features on the total of those extracted.

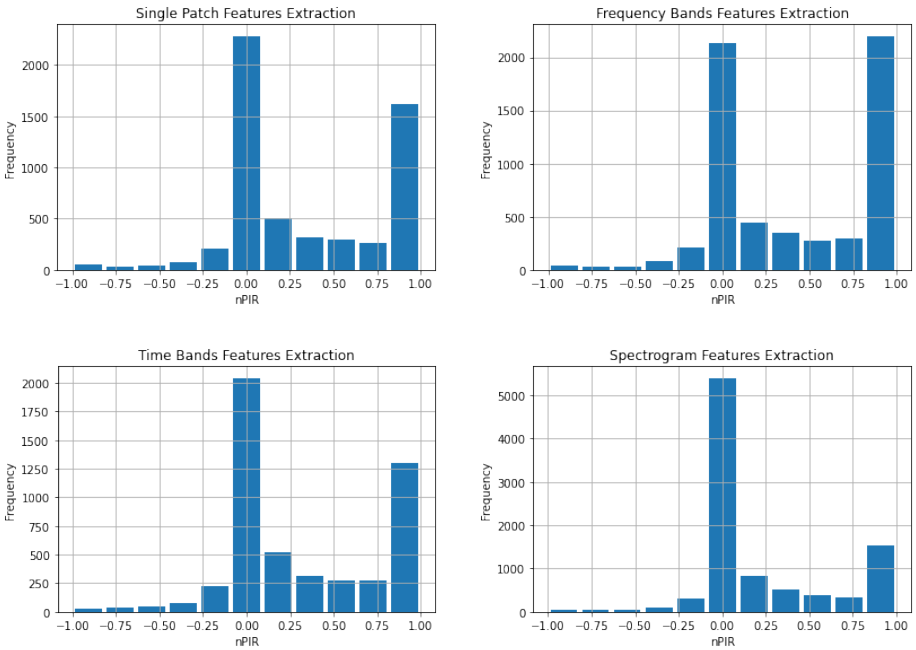


Figure 5.31: Distributions nPIR values for Features Extraction method

FEATURES EXTRACTION TECHNIQUE	PERCENTAGE POSITIVE FEATURES
Single Patch	37%
Frequency Bands	44%
Time Bands	35%
Spectrogram	22%

Table 5.7: Percentage of features with nPIR >= 0.5

Computational Cost Measuring how much time is required to complete the analysis described in the previous section we can compare the computational cost of the four features extraction techniques.
It was found that analysis using *Single Patch Features Extraction* and *Spectrogram Features Extraction* techniques takes 125 minutes over 200 input audio with an *computational time* equal to $38 \pm 2s$ for each explanation, while analysis using *Time Bands Features Extraction* and *Frequency Bands Features Extraction* methods takes 60 minutes with an computational time for each input of $21 \pm 2s$.

It is not surprising that the two methods that are based on the *hypercolums* clustering took double the time of the other two techniques. What is important to notice is that *Spectrogram Features Extraction* techniques does not lead to higher computational cost in time. This is another point in favor of preferring *Spectrogram Features Extraction* technique over *Single Patch Features Extraction* methods. It produce more precise and detailed explanations at the same computational cost.

Chapter 6

Conclusion and Future work

The objective of this thesis is to propose and test a methodology for the explanation of black-box models in the context of *audio classification*. The needs of this works is emerged looking at the lack of explanations technique in the audio classification field.

Our technique is called A-EBAnO and is based on an perturbation process over extracted interpretable features from the input, in order to measure the impact in terms of influence and precision of the perturbed features over the original predicted label.

Four different methods of features extraction are proposed each of which is focused on the analysis of different proprieties of the input.
Single Patch Features Extraction is based on an unsupervised clustering of the weights of the convolutional layers of the analyzed model. Its major objective is to highlights portions of the input that have positive, neutral or negative impact over the classification. It's usefull to understand what property of the input, in our work Log-Mel Spectrogram of the audio, is more influential.
One problem of this features extraction technique is that analyze one patch of the Log-Mel spectrogram at time, loosing the continuity in time of the input. Our answer to this problem is the *Spectrogram Features Extraction* technique. The process is the same but its done over the complete set of weights of the input, collecting the processed data from all the patches and analyze it at the same time. Comparing this two techniques on a good number of input we can conclude that *Spectrogram Features Extraction techniques* is preferable because it is capable to find a number of features that better describe the input spectrogram with a computational cost equal to the *Single Patch Features Extraction* method. The other two techniques, *Frequency Bands* and *Time Bands*, are usefull to better focus our analysis on this two aspects and experiments show how that they are very capable to underline time and frequency bands that are used by the model for the

classification.
The explanations are based on a *visual* and *numerical* part. Combining this two types of explanations we have a very intuitive way to understand the impact of features on the classification result.

Producing human understandable visual explanations was the most difficult challenge to overcome due the peculiar process of analyzing the input by the black-box model used in our project and a lot of effort has been made in the post-processing phase of the methodology in order to have result that can be human understandable.

In this work we analyzed audio that represented *environmental sound*. A future analysis of our proposed technique can be done on more task specific input as speech or music.

A first improvement of A-EBAnO will be to make it able to produce not only *local explanations* of singles input but also *global explanations* to understand the influence of specific concept on the complete set of predictions provided by the model.

The perturbation used in the project is a additive noise directly on the image of the patch of the Log-Mel spectrogram. Another solution is to perturb the audio input data and produce a perturbed Log-Mel spectrogram.

This project used a audio classifier based upon *VGGish*. The next step is first to test our methodology using other networks and to made A-EBAnO fully *model agnostic* technique.

One of the biggest improvements will be making A-EBAnO able to produce near real-time explanations of the Log-Mel spectrogram input, using algorithms and features extraction techniques based on the analysis of data *streaming* instead of the current *batch* analysis of the input, that is clearly noticeable in the current implementation explanations.

In conclusion A-EBAnO show a good capability in producing human understandable explanations of local predictions over audio classification. This is a first work that combined together eXplainable Artificial Intelligence and audio classification and so mostly of the implementation choices could be improved in future. The hope is to have being able to show that explanations on *audio classification* are possible and they can be analyzed by an human being.

Over the years Machine Learning will continue to grow and probably the model

will became more and more difficult to understand so it is very important that researches in the eXplainable Artificial Intelligence proceed and the same speed in order to produce model highly *trustable* by humans.

Bibliography

[1] A. Adadi and M. Berrada. «Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)». In: *IEEE Access* 6 (2018), pp. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052 (cit. on p. 5).

[2] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. «A Survey Of Methods For Explaining Black Box Models». In: *CoRR* abs/1802.01933 (2018). arXiv: 1802.01933. URL: http://arxiv.org/abs/1802.01933 (cit. on p. 5).

[3] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. «Show, Attend and Tell: Neural Image Caption Generation with Visual Attention». In: *CoRR* abs/1502.03044 (2015). arXiv: 1502.03044. URL: http://arxiv.org/abs/1502.03044 (cit. on p. 9).

[4] Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. «Rationalizing Neural Predictions». In: *CoRR* abs/1606.04155 (2016). arXiv: 1606.04155. URL: http://arxiv.org/abs/1606.04155 (cit. on p. 9).

[5] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. «Learning Deep Features for Discriminative Localization». In: *CoRR* abs/1512.04150 (2015). arXiv: 1512.04150. URL: http://arxiv.org/abs/1512.04150 (cit. on p. 10).

[6] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. «Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization». In: *CoRR* abs/1610.02391 (2016). arXiv: 1610.02391. URL: http://arxiv.org/abs/1610.02391 (cit. on pp. 10, 11).

[7] Ruth Fong and Andrea Vedaldi. «Interpretable Explanations of Black Boxes by Meaningful Perturbation». In: *CoRR* abs/1704.03296 (2017). arXiv: 1704.03296. URL: http://arxiv.org/abs/1704.03296 (cit. on pp. 11, 12).

[8] 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. URL: http://arxiv.org/abs/1602.04938 (cit. on pp. 12, 13).

[9] A. Dandashi and J. AlJaam. «A Survey on Audio Content-Based Classification». In: *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2017, pp. 408–413. DOI: 10.1109/CSCI.2017.69 (cit. on p. 14).

[10] S. Chachada and C. -. J. Kuo. «Environmental sound recognition: A survey». In: *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. 2013, pp. 1–9. DOI: 10.1109/APSIPA.2013.6694338 (cit. on pp. 14, 17).

[11] Jyotismita Chaki. «Pattern analysis based acoustic signal processing: a survey of the state-of-art». In: *International Journal of Speech Technology* (Feb. 2020). DOI: 10.1007/s10772-020-09681-3 (cit. on pp. 14, 15, 17).

[12] Dalibor Mitrovic, Matthias Zeppelzauer, and Christian Breiteneder. «Features for Content-Based Audio Retrieval.» In: *Advances in Computers* 78 (Jan. 2010), pp. 71–150 (cit. on pp. 15, 16).

[13] Jakob Abeßer. «A Review of Deep Learning Based Methods for Acoustic Scene Classification». In: *Applied Sciences* 10 (Mar. 2020). DOI: 10.3390/app10062020 (cit. on pp. 18, 19).

[14] Shawn Hershey et al. «CNN Architectures for Large-Scale Audio Classification». In: *CoRR* abs/1609.09430 (2016). arXiv: 1609.09430. URL: http://arxiv.org/abs/1609.09430 (cit. on p. 21).

[15] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV] (cit. on pp. 21, 22).

[16] Karol J. Piczak. «ESC: Dataset for Environmental Sound Classification». In: *Proceedings of the 23rd ACM international conference on Multimedia* (2015) (cit. on p. 25).

[17] Justin Salamon, C. Jacoby, and Juan Pablo Bello. «A Dataset and Taxonomy for Urban Sound Research». In: *Proceedings of the 22nd ACM international conference on Multimedia* (2014) (cit. on p. 27).

[18] Francesco Ventura and Tania Cerquitelli. «What's in the box? Explaining the black-box model through an evaluation of its interpretable features». In: *CoRR* abs/1908.04348 (2019). arXiv: 1908.04348. URL: http://arxiv.org/abs/1908.04348 (cit. on p. 31).

[19] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. «Hypercolumns for Object Segmentation and Fine-grained Localization». In: *CoRR* abs/1411.5752 (2014). arXiv: 1411.5752. URL: http://arxiv.org/abs/1411.5752 (cit. on p. 32).

[20] Brian McFee et al. *librosa/librosa: 0.8.0*. Version 0.8.0. July 2020. DOI: 10.5281/zenodo.3955228. URL: https://doi.org/10.5281/zenodo.3955228 (cit. on p. 50).