UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA IN DATA SCIENCE AND BUSINESS INFORMATICS

TESI DI LAUREA

Explaining Siamese Networks in Few-Shot Learning for Audio Data

Relatore
Dott. Riccardo Guidotti
Controrelatore
Prof. Giorgio Ghelli

Candidato
Andrea Fedele

Anno Accademico 2020/2021

Abstract

Traditional Machine Learning models are not able to generalize correctly when queried on samples belonging to class distributions that were never seen during training. This is a critical issue, since real world application might need to quickly adapt without the necessity of re-training. To overcome these limitations, *few-shot* learning frameworks have been proposed and their applicability has been studied widely for computer vision tasks. *Siamese Networks* learn pairs similarity in form of a metric that can be easily extended on new unseen classes. Unfortunately, the biggest downside of such systems is the lack of explainability. In this thesis we verify the applicability of Siamese Networks in the context of few-shot learning for audio inputs and we propose a novel method to explain their outcomes and assess their robustness. This objective is pursued through a *perturbation-based* method that quantifies how each input feature contributes to the final outcome by measuring the changes in the mean prediction when such feature is perturbed. We conduct several experiments on two distinct dataset to assess the method ability to explain Siamese Networks outcomes in a *C-way one-shot* framework. Qualitative and quantitative results demonstrate that our method is able to show common intra-class characteristics and erroneous reliance on silent sections. Classification weaknesses get also uncovered when audio clips are generated from heterogeneous sources and recorded in different environments.

Keywords: One-shot Learning; Siamese Neural Networks; Explainable Artificial Intelligence; Perturbation Technique; Audio; Sound Classification

Contents

1 Introduction

In recent years, Artificial Intelligence significantly sped up its pace thanks to the availability of large datasets, the emergence of powerful computing devices and the development of sophisticated algorithms (Haenlein and Kaplan 2019). Deep Learning models have been widely employed achieving powerful results in different fields. In computer vision, for example, convolution based architectures have been utilized for document recognition (Lecun et al. 1998), image classification (Cireşan, Meier, and Schmidhuber 2012) and many other tasks. Despite their thriving achievements, these systems rely on learning from big and large-scale dataset while real world applications typically involve constraints that might lead to a limited fuel of samples. In some cases the availability of training data might be limited by technical issues, while ethical or privacy issue might restrict the data collection in other circumstances. But more importantly, current techniques fail when asked to rapidly generalize from only few samples: a traditional deep learning system cannot broaden its knowledge outside the scope of the data it was trained on. On the contrary, humans are capable of learning by quickly generalizing on their prior knowledge. For example, if a child is presented with few pictures of a person he has never seen before, he will still be able to match and identify the right individual among a reasonable number of pictures portraying different subjects.

To overcome these limitations, recent studies proposed *few-shot learning* frameworks where a classifier must learn unseen classes given only few samples of each (Yaqing Wang et al. 2020). Traditionally, few-shot methods consider a *C-way k-shot* classification task where C is fixed and represents the unique class labels the model is asked to learn from while k is the number of labeled samples per each of these classes. Such parameters define the *support set* dimension, which is an auxiliary set of data the classifier is provided with to be guided in its decision. Few-shot learning aims to predict the correct class of an instance when a small number of examples of that specific class are available in the training dataset. One-shot learning is a special case of few-shot learning where exactly one example of the class is present in the training set, while zero-shot learning aims to predict the correct class without being previously exposed to any instances belonging to that class. The idea behind zero-shot learning with $k = 1$ *support sets* is learning to classify an unseen sample by using only a single labelled sample of that specific class. Different algorithms have been proposed to tackle few-shot metric-learning including Siamese Networks (Koch, Zemel, Salakhutdinov, et al. 2015), Matching Networks (Vinyals et al. 2016), Relation Networks (Sung et al. 2018) and Prototypical Networks (Snell, Swersky, and Zemel 2017). These algorithms compare inputs similarity by learning a metric during the training phase. Such metric is later extended to measure similarity between samples that were not present in the training set. Siamese Neural Networks learn a similarity measure between two inputs feeding them into two identical sub-networks that work as encoders. The inputs embedded representation are later compared by means of a distance function, which then produces a similarity metric output. Such approach to one-shot learning, with the usage of Convolutional Neural Network, has been introduced by Koch, Zemel, Salakhutdinov, et al. 2015. The capability of such networks, intrigued scientists to verify its robustness on audio inputs. Vélez, Rascon, and Fuentes-Pineda 2018 developed a Siamese Network system paired with a database which has proven to be powerful in verifying and learning new speakers by updating the database so not to retrain the network. More recently, Honka 2019 showed that a Siamese Network architecture can be employed in the context of C-way one-shot learning using environmental audio as network inputs. These works demonstrated that Siamese Networks work well with audio inputs, producing good results when asked to generalize on classes they have never seen before.

Unfortunately, the biggest downside of such systems is the lack of explainability. It is hard to explain and interpret how these architectures can mimic human behaviour when they correctly scale up on unseen samples. The need of explainability has become more and more important as Deep Learning network became popular and widely deployed. Understanding the reason why a models take a specific decision is crucial to developers, organizations and moreover to the end users on which such decisions fall upon. While traditional Artificial Intelligence systems might need less effort to be explained, Deep Learning models are often compared to black boxes where it is unclear how and why a given input triggers a specific output. In recent years, different researches examined the reason behind the need of explanations and the social impact of how people select, evaluate and communicate them (Miller 2017), while other studies targeted the problem of explaining black-box models (Guidotti et al. 2018, Adadi and Berrada 2018). One of the possible partitions of these techniques considers *gradient-based* and *perturbation-based* methods. While both partitions aim to understand the role that each input feature plays on a specific output, they solve the problem differently. The former are faster, given that they estimate feature attribution values by means of forward and backward propagation passes throughout the network. On the other hand, occlusion-based methodologies perturb the input and measure how the output changes with respect to the original input. Techniques belonging to the latter family require to test sequentially each feature (or group of features) resulting in longer computational times. Different explanation methods have found a very good feedback in the research community for black-boxes, but few techniques have been presented to interpret and explain Siamese Networks. Moreover, to the best of our knowledge, none of the available techniques has ever been tested on audio inputs processed by such neural networks in the context of *C-way one-shot learning.*

In this thesis we propose a novel perturbation-based method to explain Siamese Networks in the context of *C-way one-shot* learning on audio input data. Our proposal seeks to understand what are the discriminative feature for a machine learning model that is asked to quickly learn and generalize as humans do. We want our method to answer the informal question that asks *What is the model listening to when it correctly matches the class of two audio it has never heard before*? But also *why a given recording is more similar to a specific audio more than it is to others*? And finally *why is the model miss-classifying a given audio*? The method we introduce uses a perturbation approach evaluating *segment-weighted-average* contribution values to the final outcome considering the interplay between different areas of the input as a whole. The contribution values can then be visualized as heatmaps to have a quick and easy-to-understand idea of the network classification behaviour. Combining the visual communicative effect of such heatmaps and the soundification of the most important segments, our method is able to guide us in understanding both correct and incorrect classification outcomes. Experiments demonstrate that Siamese Networks reach state-of-the-art performance on both short and longer clips in the context of *5-way one-shot* learning. Moreover, experimental results of the proposed explanation method illustrate that the robustness of such networks is very much tied to the recordings domain. Class-homogeneous dataset seem to lead to robust networks, while class-heterogeneous ones result in more inaccurate classification behaviours. Our method also brings to light an erroneous reliance on silenced areas that, in some cases, is the cause of miss-classification errors.

This thesis is divided as follows: in Chapter 2 we review the state-of-the-art literature about the usual employment of Siamese Networks, the explainability techniques used on audio data and the available approaches to explain Siamese Networks. We then provide background information on the central topics of Siamese Networks and C-way k-shot learning in Chapter 3, concluding with an exhaustive discussion of Explainable Artificial Intelligence (XAI). In Chapter 4 we formalize the problem we seek to address, while in Chapter 5 we give an in-depth description of the perturbation-based explanation method we propose. Finally, in Chapter 6 we report the experiments we conducted to validate the tool and we conclude summarizing the experimental results and future research directions in Chapter 7.

2 Related Work

In this chapter, we present the state-of-the-art researches in which our project sets the ground. In Section 2.1 we start by introducing the employment of Siamese Neural Networks in the context of few-shot learning. We continue in section 2.2 presenting latest explainaiblity approaches on traditional neural network architectures for audio and time-series inputs. In Section 2.3, we conclude illustrating recent methods proposed to specifically explain Siamese Neural Networks.

2.1 Siamese Networks in Few-shot Learning

Koch, Zemel, Salakhutdinov, et al. 2015 introduced the use of Siamese Networks for Image Recognition in the *one-shot learning* framework. In this work, the network is implemented as a convolutional neural network and it is trained to rank similarity between inputs. After training, the tuned similarity function is used to predict on new classes from unknown distributions. The Ominglot Dataset[1] is used to train the architecture: it is composed of 50 alphabets, containing from about 15 to 40 different characters each. 20 different drawers produced an instance of each of these character across all alphabets. 40 of the 50 alphabets are referred as a *background set*, which is used to train the network similarity function in measuring the distance between character-pairs. The remaining 10 alphabets are referred as *evaluation set*, which is only used to measure the one-shot classification performance. These terms are carefully chosen by the authors and they differ from the traditional so called training, validation and test set. During the training phase, the background set is exclusively used to train the feature similarity function, while the evaluation set is used to establish forty 10-way one-shot classification task per each of its alphabet, resulting in a total of 400 one-shot validating trials. In each task, standard classification accuracy is computed considering correct a classification which indicates as most similar two instances of the same character. The authors showed that such

[1]`https://github.com/brendenlake/omniglot`

framework could reach human-level accuracy, opening the usage of this approach to other domain.

Such suggestion was well welcomed by Honka 2019, where a Siamese Network was framed in a C-way one-shot framework using environmental audio as inputs. The authors used the ESC-50 dataset[2] containing a collection of 2000 annotated 5 seconds audio clips belonging to 50 different classes. In this work, a convolutional neural network architecture was trained on 40 classes to learn a similarity function, while 5 separate classes were used to evaluate 400 randomized 5-way one-shot tasks accuracy. An interesting difference was introduced in this research since the 5 remaining classes were used in order to test the one-shot model performance. The validation and test set were kept separate in order to ensure that the training process would not only lead to hyperparameter optimization on the validation set, but to unseen samples as well. A final mean accuracy of **79.1%** was obtained, showing that such framework can be used on audio inputs with reasonably good results.

The ESC-50 dataset was also used by Chou et al. 2019 in the *C-way k-shot* framework to compare different metric-based algorithms. The authors propose a novel attention similarity function that computes the *attentional weighted average of segment-by-segment* similarity between vectors. The concept of attention is introduced to guide the model in paying attention to specific segment of actual sound events in longer recordings clips. Experiments show that the tested algorithms performance increase when using such attention function indeed. Recalling that C is the number of different classes on which the model is queried on and k represents the number of samples for each of these classes, it is worth summarising some other findings. Learning tasks on $k = 1$ perform worse than $k = 5$ for both $C = 5$ and $C = 10$, suggesting that having more samples per each class smooths the model correct classifications. Finally, it is illustrated that $C = 5$ perform worse than $C = 10$, suggesting that choosing the right class among a higher number of overall labels is a harder task.

In a recent research Vélez, Rascon, and Fuentes-Pineda 2018 used a Siamese

[2]`https://github.com/karolpiczak/ESC-50`

Network architecture for a speaker recognition task, introducing a database where known speakers' sample are stored. Every time somebody speaks to an input microphone, such sample is matched with every of the current stored labels. The speaker resulting in higher similarity is considered to be the same of the one that produced the input but, if no stored speaker matches the input, a new entry is added to the database for further uses. Validation and test phases are carried out in a traditional manner: the authors do not monitor the mean accuracy of a *C-way k-shot* learning task. Experimental results show Accuracy, Precision and F1 scores ranging from **89%** to **98%** for the three proposed models when queried on 500 samples belonging to the same speaker and 500 belonging to different ones. Moreover, the article shows how the model performances improve when the number of stored samples per each speaker is increased from 1 to 10. Such finding is nothing new: it suggest that a higher value of *k* samples to be compared with a new query helps the model in matching them correctly.

Our project follows the general approach proposed by Koch, Zemel, Salakhutdinov, et al. 2015 and Honka 2019, where Siamese Networks are trained to learn pairs similarity and later queried in a *C-way one-shot* learning framework on samples belonging to unseen class distributions. Differently from the last research we described, we measure the models training performances by means of classical Accuracy, Precision and F1 scores on training batches and we test them in terms of *mean accuracy* on several random one-shot tasks. We also study such mean accuracy behaviour by increasing C similarly to what Chou et al. 2019 and Vélez, Rascon, and Fuentes-Pineda 2018 do.

2.2 Audio and Time Series Explainability

Becker et al. 2018 approached the problem of explaining a black-box convolutional based architecture which classifies audio signals. The authors introduce a novel audio dataset of English spoken digits, which they refer to as AudioMNIST[3], containing 30000 recordings of spoken digits (from 0 to 9) with 50 repetitions per digit for each of the 60 different speakers. Two different architectures are trained on

[3]`https://github.com/soerenab/AudioMNIST`

audio's spectrograms and waveforms respectively. Both networks, with the obvious changes in the last layer dimension, are deployed in the digit and speakers' gender recognition tasks reaching a mean accuracy that range from 91.74% to 95.87% with a standard deviation of ±8.6% and ±2.85% respectively. The *Layer-wise Relevance Propagation* (LRP) technique introduced by Bach et al. 2015 was employed to back-propagate the hidden layers' neurons relevance score from the output towards the input. Visualizing the relevance score obtained, the authors show how the network would focus on different areas when asked to recognize digits. Such visualization technique lacks unfortunately of interpretability: it is almost impossible to map such features to higher concepts like, for example, phonemes. The relevance score obtained in the speaker' gender recognition task lead to the hypothesis that the network is looking at the fundamental frequencies and their immediate harmonics, which are known to be in fact discriminant features in genders (Traunmüller and Eriksson 1995). Finally, the authors assess that for both tasks the network indeed relies on those samples found to be relevant by LRP: the accuracy decrease drastically to 25% and 50% when 20% of those features are set to zero in digit classification and gender recognition respectively.

A local interpretable model-agnostic explanation method for music content analysis was introduced by Mishra, Sturm, and Dixon 2017. The authors propose *SLIME (Sound-LIME)*, which is based on the most famous *LIME* (Ribeiro, Singh, and Guestrin 2016) and offers 3 different explanations based on the prior audio segmentation granularity: temporal, frequency and time-frequency segmentation. The explainer goal is to find those *super-samples* that influence the classification the most when turned on/off. Similarly to LIME, synthetic samples close to each of the 3 segmented representations are generated, by turning on/off specific super-samples. Each generated sample is then weighted according to its proximity to the original audio. Finally, a linear model is employed to approximate the black box classification based on the synthetic samples and their weights. The article compares SLIME explanations both on a Decision Tree and a Random Forest that classify whether or not an audio contains a singing voice. Despite a good overall accuracy of 71% the SLIME technique highlights that the super-sample considered to be important for the Decision Tree are mainly instrumental, proving that such method can indeed display when a model is not reliable. The article finally studies a convolution based architecture by means of the gradient-based *Saliency Map* technique (Simonyan, Vedaldi, and Zisserman 2013) obtained back-propagating the output towards the input using *leaky ReLU* functions so to prevent vanishing gradient. The super-sample obtained by such technique are compared to those considered to be important by SLIME and a match of 46,5% is described on 1349 randomly selected audio.

An interesting study on Time-Series was presented by Tang, Liu, and Long 2020, where the authors introduce the *Dual Prototypical Shapelet Network (DPSN)* few-shot classification framework that opens to two different level of interprenability: a *global overview* by means of representative shapelets, and *local highlights* using discriminative shapelet. While the former pools together common aspects of samples belonging to the same class, the latter show discriminative aspects between different classes. The presented framework consists of 3 main components: the feature extraction, which transforms the input time-series into a Symbolic Fourier Approximation (SPA), the classifier that builds up classes prototypes by means of a metric-learning neural network and finally the shapelet feature extraction from the initial time-series. By construction the classifier is able to find *representative shapelet*, while the combination of shapelet and prototype's feature recovers *discriminative shapelet*. The prototype network is a KNN based classifier, which classifies samples of the class with the nearest prototype. The authors compare the proposed *DPSN* on 18 different dataset, outperforming the state-of-the-art algorithms in almost all of them. The article concludes by showing some graphic examples of class discriminative shapelets for some of these datasets.

Our research tend to emulate the way input are pre-processed similarly to what Becker et al. 2018 and Mishra, Sturm, and Dixon 2017 do by treating audio data via their spectrogram representation, with the due differences in terms of parameters configuration. Differently from the first research we do not employ any *gradient-based* explanation method, but a *perturbation-based* one. Moreover, we do not make use of any additional model like Mishra, Sturm, and Dixon 2017 do, and we propose to use a different approach to segment spectrograms. Finally, it is important to notice that we do not treat inputs as time series like Tang, Liu, and Long 2020 do due to much higher sample-rate of the audio recordings.

2.3 Siamese Networks Explainability

The first research worth mentioning explores Siamese Convolutional Neural Network in a query-by-vocal-imitation sound retrieval system (Zhang, Pardo, and Duan 2019). The architecture compares a vocal imitation to the actual sound recordings stored in a database to find the most similar one. In addition to standard architectures, the authors propose *Semi-Siamese* Networks where sub-networks architectures and weights are partially shared between the two network branches. Experiments consider also the possibility of pre-training each network's branch on different tasks, resulting in the outperformance of current state-of-the-art. The last section of the article focus on visualizing and sonifying the input patterns that maximize the activation of certain neurons, using the activation maximization approach (Erhan, Courville, and Bengio 2010). Such approach keeps the network trained weights unchanged by gradient ascent a specific neuron's activation with respect to the input from different random initialization. After convergence, the updated input spectrogram can be considered as what the neuron learns. The avoid the vanishing gradient trap, ReLU function are replaced by *leaky ReLU* ones with a slope of 0.3 for negative inputs. The recovered areas get later sonified recovering the phase information thanks to the *Griffin-Lim* algorithm (Griffin and Lim 1984). Visualizing the patterns that maximally activate random neurons from each convolutional layer, it is possible to have a glimpse of what is believed to be important and its overall magnitude. While top layers' neurons seem to learn local features, deeper layer's neurons are likely to capture spectrogram-like patterns of different frequency ranges. Similarly, the soundification of top layers' neurons seem to focus on low frequencies that progressively capture spectral components until grasping birds chirping and water flowing like sounds in deeper layers. Unfortunately, this technique considers random neurons living room for the isolated effect it might cause to the network overall job. In addition, it is not possible to comprehend whether the return features have a positive or negative impact on the similarity prediction. Moreover, the encoders are considered as stand-alone and their effect on the Siamese Network's core distance layer is bypassed.

A similar approach to explain Siamese Networks is briefly described by Acconcjaioco and Ntalampiras 2021, where the system is developed in the context of one-shot learning to identify bird species in non-stationary environments. The main goal is to build a system able to identify an unseen sample of a known class and to store new classes when received. The mean accuracy of a *C-way one-shot* task is not measured neither during training nor during evaluation, but experimental results show that the network reaches **70.7%** accuracy with a standard deviation of $\pm 8.09\%$ when tested on 5 unknown classes (free-from-noise dataset[4]) and **72.36%** accuracy with a standard deviation of $\pm 0.67\%$ on 3 unknown classes (non-stationary environment dataset[5]). While the former result seems to be aligned on usual *5-way one-shot* tasks, the latter seems to under-perform if compared to usual *3-way one-shot* learning tasks. Finally the authors visualize how spectrograms are decomposed by each convolutional layer, observing that they each simplify the input and focus on specific regions. Among all layers, they tend to consider the last one as the explainer concluding with the hypothesis that the most distinctive feature is the distribution of the signal's energy in species-dependend frequency bands. It is important to notice that the explanation is derived only from the network's encoders, without considering the Siamese merging distance layer.

The current only proposal to explain Siamese Neural Network is introduced by Utkin, Kovalev, and Kasimov 2020, where a special auto-encoder is proposed as core to the explaination. First, the encoder-part is trained to reconstruct the input instances of the training set starting from the embedded representation given by the Siamese Network's embedder. The loss function used during this training procedure takes into account proximity of the network's embedded vectors and the encoder-part hidden representation. Then, the decoder-part is trained to reconstruct the original input given the hidden representation resulting from the encoder-part. Both encoder and decoder can be further trained on samples that don't belong to the training set.

[4]`https://xeno-canto.org/about/xeno-canto`
[5]`https://zenodo.org/record/1250690`

Once the auto-encoder is trained, a pair of input to explain is given as input both to the Siamese Network and the auto-encoder itself. The vectors resulting from the Siamese Network's encoders, gets perturbated on what the authors refer to as *important features.* Such features are chosen considering the smallest distance if the two input are semantically close, while the biggest distance is considered otherwise. The resulting perturbed vectors are passed to the decoder-part, that map them back to the original input space. The mean attribution value of each feature is measured by randomly perturbating the embedded vectors considering the difference between that feature value in the reconstructed input after perturbation and its value in the reconstructed input without perturbation. Empirical results on the MNIST dataset[6] show that this technique is indeed able to bring to life features that highlight input semantic similarities/dissimilarities. Limitations of this approach are the large number of tuning parameters, including the substantial choice of the *important features'* number, and the need of a large number of data to train the auto-encoder on. Moreover, this algorithm requires to train an additional model that needs to have access to the training set. It is important to observe that the Siamese Network's distance function and its associated similarity score are bypassed. The embedded vectors are subjected to a perturbation approach, which later computes the difference between the input that the external decoder reconstructs with and without perturbation.

The explanation methodology we propose in this thesis differs from the ones employed by Zhang, Pardo, and Duan 2019 and Acconcjaioco and Ntalampiras 2021 since they use *gradient-based* methods that tend to focus only on the encoder part of the system. Both these researches limit their exploration to the last convolutional layer of the network not considering the architecture as a whole. In addition to this, the first research only considers random neurons at random layers of such encoders. Differently from what Utkin, Kovalev, and Kasimov 2020 propose, we do not train any external model. Moreover, our explanation method does not need access to the training data and can be deployed directly at prediction time. The main goal of our project is to explain Siamese Networks in their entirety, focusing on every layer of the architecture and especially the ones responsible for the final similarity

[6]`http://yann.lecun.com/exdb/mnist/`

score prediction. Moreover, our proposal wants to address the problem of explaining siamese networks in the context of *C-way one-shot* learning which is not tackled by any of these works.

3 Background

This chapter provides background information on topics considered to be essentials in the context of our project. In Section 3.1 we introduce Siamese Networks, while Section 3.2 describes the project framework of C-way k-shot learning. Finally, in Section 3.3 we describe the importance of Explainability and the most renowned techniques employed for black-box models.

3.1 Siamese Networks

A Siamese Neural Network is a neural network composed of two identical sub-networks, which have the same architecture and share the same weights and parameters configuration. The weights updating process is mirrored across the branches, so to have two identical encoding networks. The main idea is to input two samples, one per each twin branch, and train a *pairwise similarity* function that will output 1 if the two samples are somehow similar (e.g. belong to the same class) and 0 otherwise. The sub-networks function as encoding layers, mapping inputs into an embedding space where a distance function is later employed to calculate the distance between the embedded vectors. Based on the distance, a similarity score in the range of *[0, 1]* is finally computed. This paradigm was first introduced by Bromley et al. 1993, where a Siamese Network is used to compare known signature to unknown ones. The applicability range of this kind of network is broad considering that it learns semantic similarities between the inputs, and it may vary from face recognition to spam detection. The overall training phase considers input pairs (x_i, x_j) where x_i and x_j belong to the input domain (e.g., images), and assignes a a binary label $k_{i_j} \in \{0, 1\}$ with value 1 if x_i and x_j are semantically similar, and 0 otherwise. A non-linear embedding function f is applied to each input by the respective branch taking the input to some n-dimensional Euclidean space such that $f(x_i) \rightarrow h_i$, $f(x_j) \rightarrow h_j$ where h_i and $h_j \in R^n$. The distance $d(h_i, h_j)$ between embedded vectors should be as small as possible for a pair of instances with $k_{i_j} = 1$,

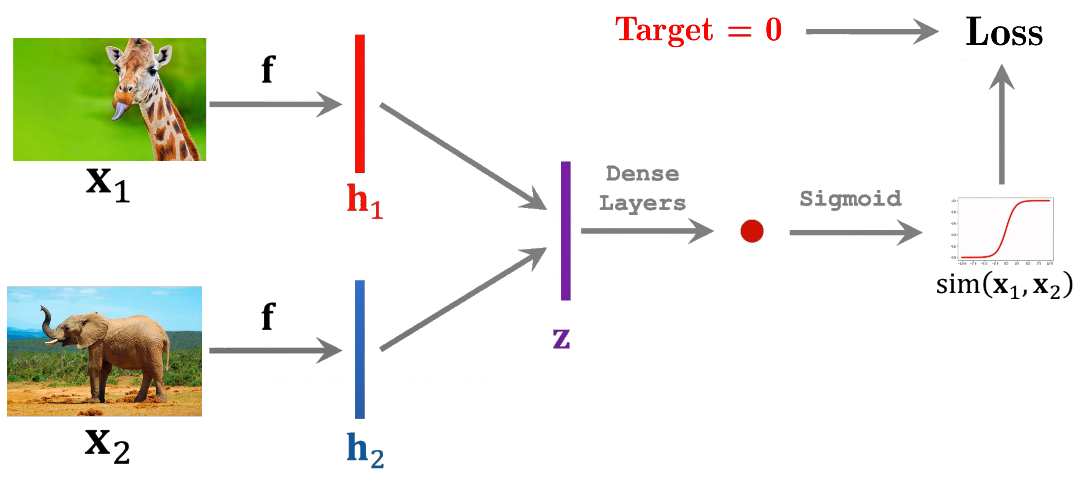


Figure 3.1 *Example of a Siamese Network that process images and uses the sigmoid activation function to compute the final similarity score.*

while it should be as large as possible otherwise. The distance vector $z_{i_j} = d(h_i, h_j)$ can be further processed and finally passed into an activation function to output a similarity score $s_{i_j} \in [0, 1]$. Considering the target label k_{i_j} and the predicted similarity score s_{i_j}, a loss function is used to measure their difference and to back-propagate the gradients to update the model weights. An example of a Siamese Network is depicted in Figure 3.1.

The actual implementation of these network may vary in different aspects according to the goal. First of, the dataset split could be approached in a traditional way: train, test and validation set are composed of samples belonging to disjoint sets, but from a known distribution. A different approach would imply a split where the validation, the test, or both sets contain samples of unknown classes since the similarity function does not care about the data distribution but mostly their similarity. Another implementation difference might be in the distance function used to compute the distance between the embedded vectors like, for example, *Manhattan or Euclidean* distance. The loss functions used to optimize the weights while training may vary as well. Our project focuses on three functions, the first one being *binary cross-entropy* defined as

$$L_{BCE} = -ylog(p) + (1 - y)log(1 - p)) \tag{3.1}$$

where p is the predicted value and y the input pair label. This loss function is widely used for classification tasks and it can be integrated in Siamese Networks since the output can be read as a similarity score. The second loss function is the *mean squared error (MSE)* defined as follow for a single binary output

$$L_{MSE} = (y - p)^2 \tag{3.2}$$

where p and y are again the predicted value and the input pair label respectively. The last loss function is *Contrastive Loss* (Khosla et al. 2020), which gets commonly employed in Siamese Networks since it forces similar samples in having smaller distances, while dissimilar data points result in larger distances and it is defined as

$$L(W, X, y) = (1 - y)\frac{1}{2}(D_w)^2 + (y)\frac{1}{2}\{max(0, m - D_w)\}^2 \tag{3.3}$$

where $X = (x_1, x_2)$ is the pair of inputs x_1 and x_2 and W is the set of parameters to be learnt. Here again y is the pair label (1 if similar, 0 otherwise). This formula introduces m as a margin for dissimilarity, and the euclidean distance D_w between the output of the encoding sub-networks. These three loss functions were tested in different configuration and a description of such experiments will be detailed in Chapter 6. For the sake of completeness, we want to mention the more recent *Triplet loss*, where an anchor input is compared to both a positive and a negative one. Such function minimizes the distance between the anchor and the positive example, while maximizing the distance between the anchor and the negative input (Hoffer and Ailon 2014). This function has to be incorporated in a three-branched network, one per each of the input[1].

A crucial aspect of training a Siamese Network is the criteria used to select training positive and negative examples. *Positive examples* represent those elements that are similar (e.g. belong to the same class), while *negative examples* contain dissimilar elements (e.g. belong to different classes). The number of pairs to learn from is quadratic: each instance of a specific class can be paired with each other instance of the same class and marked as positive, and paired with each instance of every

[1]`https://keras.io/examples/vision/siamese_network/`

other class and marked as negative. One could decide to use every possible pair resulting in longer computational times, or set the exact number of positive and negative pairs to be used per each class. Another limitation could be introduced by excluding pairs reuse in different train iteration. Moreover, the ratio between positive and negative pairs might vary leading to balanced or unbalanced training sets. Finally, the network inner architecture could vary. Convolution-based architectures are often employed due to their ability to extract different semantic information in each filtering layer, but LSTM-architectures are quite used as well in many tasks like, for example, text categorization (Shih et al. 2017) and disease detection (Bhati et al. 2019).

3.2 C-way k-shot Learning

The zero-shot learning framework consist of training the model on specific classes, and later query the same model with samples of unknown classes that were never seen by the model while training. To back up the model decisions, a *support set* is introduced: this set contains ***C**-way* classes and ***k**-shot* samples per each of class. One approach to few-shot learning is *metric learning*, which implicates learning an embedding space to compare classes. The goal is to learn a similarity function that given two similar input returns 1, and 0 otherwise. Once the similarity function is trained, it is used for prediction: the query sample is compared with each element of the support set. The support sample returning the highest similarity score is considered to be closer to the embedding space of the query sample, therefore classified as similar to some criteria (e.g. belonging to the same class). Figure 3.2 shows an example of a 6-way 1-shot support set and a query image. State-of-the-art researches show that the performances tend to get worse as the number of ways *(C)* gets higher, while they get better as the number of shots *(k)* gets higher (Wolters et al. 2020, Yu Wang et al. 2020). This suggests that it is easier to find the right element from the support set, when the number of overall classes to pick from is limited and it gets progressively harder when such number increases. On the other hand, if more samples per class are provided the model is better guided to the right similarity match. It is important to notice that *C-way k-shot* learning tends to

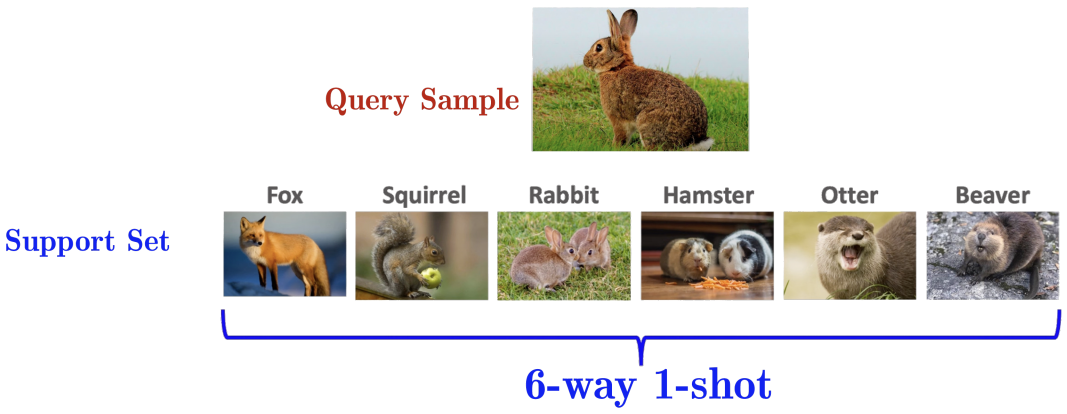


Figure *3.2 Example of a 6-way 1-shot support set containing 6 classes and 1 samples per each class and a query sample. We expect the rabbit support sample to return the highest value of similarity when compared to the query sample.*

mimic human behaviours, overcoming some real-word scenario limitations. Having few training samples becomes less problematic, since the only thing that matters is the model ability of measuring similarities and this might not directly depend on the number of samples the model is trained on. On top of that, once the model has learned how to compare similarities between inputs it can be queried on samples belonging to unknown classes as long as a support set is exhibited. It is easy to see how the Siamese Networks architecture can easily be employed in this learning framework.

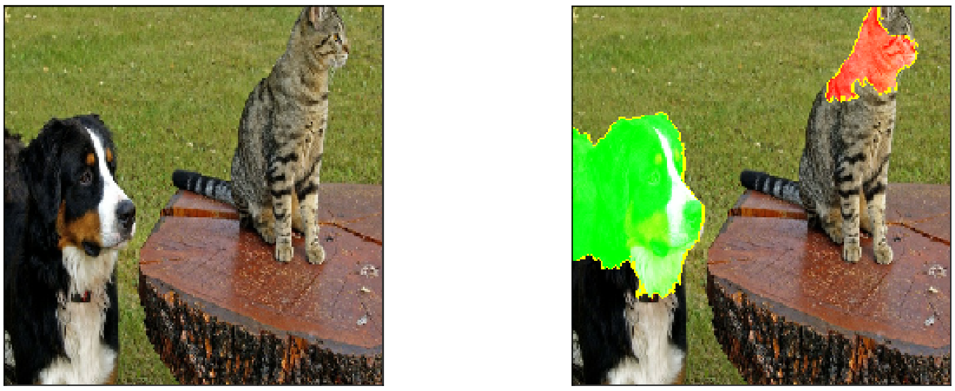
3.3 Explainable Artificial Intelligence

Explainable AI, namely XAI, is Artificial Intelligence of which results can be explained and interpreted by humans. The need of knowing the whys and hows behind a model results could be beneficial for different users and for different reasons. The end-user of an AI application, might find insightful the explanation of an outcome more than the outcome itself. On the other hand, developers could use explanations to better understand if the model is miss behaving for some reason and if there is the need of repeat some procedures in a controlled environment. Last but not least, stakeholders might want to understand why a model is taking specific decisions, before letting that model impact on real people's life. The *right to know* (Dimitrova 2020), is the right of having an explanation for a specific outcome of an algorithm and the rise of the data-driven society led to a large social discussion of why such right must be considered as an individual right. The discussion on explainable algorithms demonstrated that such information are not primarily useful to those who develop them, but to the people and the society where such algorithms are deployed in. The European Union enacted in 2016 the *General Data Protection Regulation*[2], which in *Recital 71*[3] define and characterize such *right to know*. In 2018 Google announced the *What-if Tool* (Wexler et al. 2020), a tool that visually probe the behavior of trained Machine Learning models to test performance in hypothetical situations and to analyze the importance of different data features for different fairness metrics. It goes without saying, given that tech-giants realised how important is to develop and deploy responsible Machine Learning models, that the effort of the research community has been major in the field especially when facing the black-box models transparency challenge.

LIME (Ribeiro, Singh, and Guestrin 2016) has become one of the most popular methods since it is model-agnostic and works on text, image and tabular data. It generates local explanation by creating a set of synthetic instances in the neighborhood of the instance to be explained and weights them according to their proximity. Then, a linear model is used to approximate the model in the vicinity of the instance to be explained. The main intuition behind this technique is to use a linear model to approximate and explain a more complex one. Users of different expertise can rely on this explainer because it produces interpretable representation and easy-to-understand explanations. This method is widely employed in text and image analysis as well, generating explanations in form of fragments of texts or superpixels of an image. Figure 3.3 shows an example of explanation in form of important superpixels. The success of LIME has inspired many adaptations that build up from the ground of its main idea and propose improvements on different levels like, for example, *ALIME* (Shankaranarayana and Runje 2019), *DLIME* (Zafar and Khan 2019) and *LIME-SUP* (Hu et al. 2018).

Another very popular method is *SHapley Additive exPlanations*, namely *SHAP*

[2]`https://www.privacy-regulation.eu/`
[3]`https://www.privacy-regulation.eu/en/r71.htm`



(a) Input image **(b)** LIME Explanation towards the dog class

Figure 3.3 *LIME explanation of a black-box model classifying the left picture as containing a dog. The explanation highlights in green the top 5 super-pixels that are most positive towards the class, while the red 5 top super-pixels are the most negative ones.*

(Lundberg and Lee 2017), which is model-agnostic and aims to explain a model prediction by computing the contribution values of each feature to that specific prediction. To achieve this, it optimizes a regression loss function based on the concept derived from game theory of *Shapely Values*. The idea behind SHAP is to assess which of the features has the greatest contribution to the overall prediction. The technique considers the average marginal contribution of a feature value across all possible coalitions, where each coalition vector assigns a value of 0 or 1 to each feature defining whether or not it is present in the coalition itself. SHAP generates easy-to-understand explanations that are able to highlight which features contribute the most and if their influence is positive or negative to the final outcome. Figure 3.4 shows an example of this method's explanation on image input data.

Another different group of methods goes under the umbrella of *gradient-based* techniques, which estimate features attribution values to the final outcome with forward and backward propagation iterations through the network. On the the most known technique is *Grad-CAM* (Selvaraju et al. 2019), which stands for Gradient-weighted Class Activation and it is a technique for producing visual explanations of CNN-based models. This method uses the gradients of any target concept obtained during the back propagation phase, flowing them into the final convolutional layer so to produce a coarse localization map that highlights regions believed to be important by the model. *Epsilon-LRP* (Bach et al. 2015) is another gradient based method, which computes layer-wise relevance propagation and it is able to high-

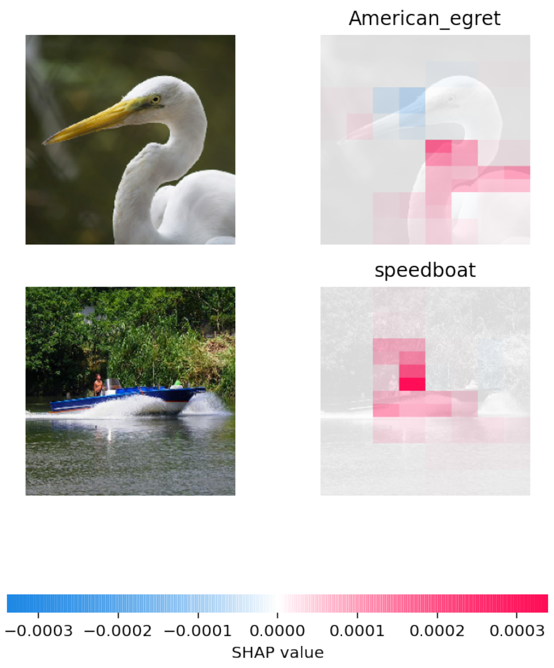


Figure 3.4 *SHAP explanation on a multi-class classification task. While for the* ***American egret*** *class the explanation finds both positive and negative influence in the body and beak areas respectively, only a positive influence is found for the* ***speedboat*** *class on the image features portraying the actual boat.*

light positive contributions to the network classification. It produces a heatmap in the input space indicating the attribution relevance of each feature to the outcome. Another technique is the one known as *Integrated Gradients (IG)* which computes the gradient of the model's prediction output to its input feature by establishing a baseline and a sequence of samples interpolated from the baseline to the actual input. *DeepLIFT* introduced by Shrikumar, Greenside, and Kundaje 2017 is also a member of the gradient-based family, and it is a method for decomposing black-boxes output prediction on specific inputs by back-propagating the contributions of all neurons to every feature of the input. The activation of each neuron is compared to its reference activation and a contribution score is assigned according to this difference. In Figure 3.5 a comparison of the described gradient-based methods is presented on various inputs from different datasets.

On the other hand, *Perturbation-based* methods assume that the features contribution can be determined by measuring how outcome scores changes when the input feature is altered. These methodologies might be slower than the gradient-

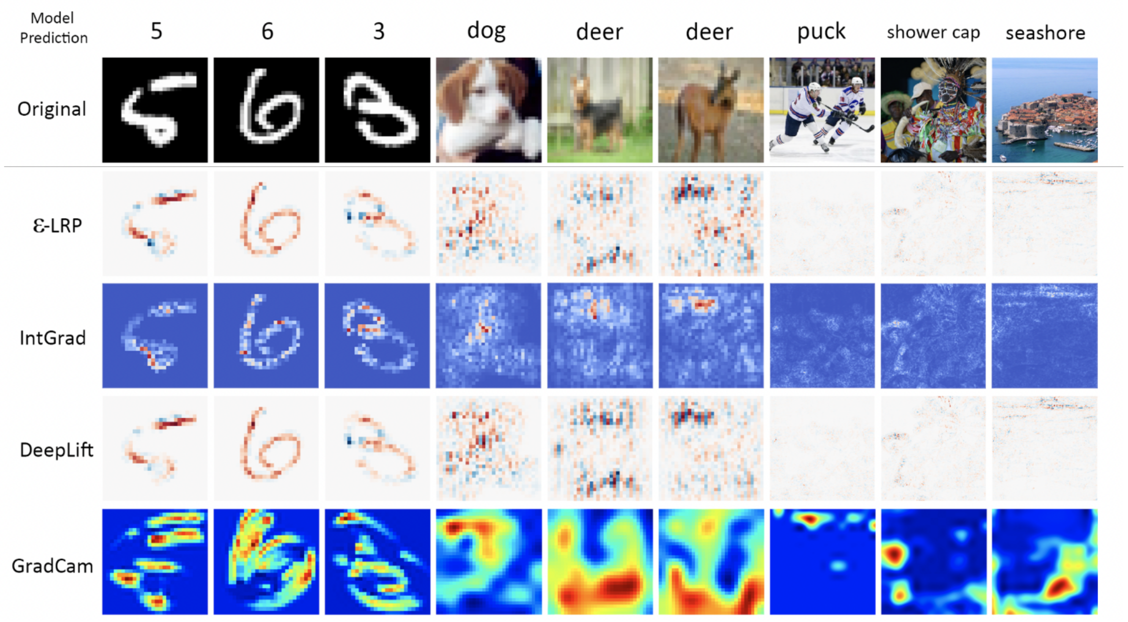


Figure 3.5 *Portion of a figure exhibited in Bodria et al. 2021 showing saliency maps obtained with different gradient-based methods on different datasets. The first row contains the original input image and the model predicted class.*

based ones, since the overall number of features (or group of features) to perturb might be large. The *Occlusion* method introduced by Zeiler and Fergus 2013 produces changes in a classifier's output by perturbing input images by sliding an empty window over them. Such window might be of different dimension, therefore resulting in different attribution values. The above-mentioned LIME (Ribeiro, Singh, and Guestrin 2016) method itself falls under this family umbrella, since it employs super-pixels occlusion to compute the most influential ones. Perturbation methods are model-agnostic by definition since the only thing they need access to is the model outcome value, so to compare it after different perturbation runs.

Different methods were proposed for both gradient-based (Ancona et al. 2018) and perturbation-based (Ivanovs, Kadikis, and Ozols 2021) families and, while the goal of this project is not to give an exhaustive list of them, we want to point out that there exist different general approaches when black-box models have to be explained. While some methods might be tied to a specific input data format, others tend to work on different input opening to tests in different contexts. Dealing with black-box models is not an easy task, but developing model-agnostic technique has found a good response in the research community. Every method works differently than the others under the hood, but they all aim to provide clear and easy-to-understand explanations. The form of such explanations is as close as possible to the human reasoning and it tends to be as graphic as possible like. Image-related explanations, for example, are usually heatmaps of the areas that play the bigger part in the model classification behaviour. Explanations have to be easy-to-read, to guide humans in understanding *what and how much* influences the black-box models final outcomes.

4 Problem Formulation

This thesis want to build ***Siamese Networks*** by training them on ***audio inputs*** in the context of ***C-way one-shot*** learning, and it wants to address the challenge of ***explain*** such models outcomes. As of today, to the best of our knowledge, few researches address the study of implementation-related details of this type of architecture specifically for audio input data. Moreover, explaining the whys behind the outcomes of Siamese Networks has never explored before for such inputs, especially in the context of *few-shot* learning. The final goal of our work is to assess whether or not Siamese Networks are able to correctly classify audio recordings belonging to class distributions that were never seen during training. More importantly, we want to answer the informal question that asks *what is the model listening to when it correctly matches the class of two audio it has never heard before*? But also *why a given recording is more similar to a specific audio more than it is to others*? And finally *why is the model miss-classifying a given audio*?

Let us formalize these two separate problems. Given:

- $\boldsymbol{C} = \{c_1, c_2, ..., c_n\}$ a set of n distinct classes;
- $\boldsymbol{S} = \{s_2, s_2, ..., s_n\}$ a support set of n samples belonging each to a specific $c_i \in C$;
- $\boldsymbol{x}$ a query sample belonging to a class $c_i \in C$;
- $\boldsymbol{Y} = \{y_2, y_2, ..., y_n\}$ a labels set where $y_i = 1$ if the class of the support sample s_i is the same of the query sample x, and $y_i = 0$ otherwise.

We define $P(c_i|x, s_i)$ as the probability that the query sample x belongs to class c_i, given the support sample s_i. We want to train a model to predict the query sample class $\hat{c}$, being this the biggest probability given the pair-wise comparison between the query sample and each element $s_i \in S$:

$$\hat{c} = \underset{\forall c_i \in C; \forall s_i \in S}{\arg\max}\ P(c_i|x, s_i) \tag{4.1}$$

and we want to do this in such a way that $y_{\hat{c}} = 1$. Ultimately, we want that the support sample s_i resulting in the highest similarity score belongs to the same c_i class that the query sample x belongs to. Let us define f as such model prediction function that given a query sample x and a support set S, returns the class $\hat{c}$ of x or, formally, $f(x, S) \rightarrow \hat{c}$.

The goal of this work is to define a function δ that, given the model prediction function f along with the query sample x and the support set S, it explains why the model classifies x as class $\hat{c}$ by means of a set $H = \{h_1, h_2, ..., h_n\}$ of n heatmaps. Such explanation function is formalized as $\delta(f, x, S) \rightarrow H$. Each h_i contains the *contribution values* expressing how each area of each support sample $s_i \in S$ contributes to the similarity score given by the model when comparing the query sample x with the support sample s_i in exam. This will allow to assess how each portion of the support sample s_i affects the model classification towards the class c_i when compared to the query x, for each $c_i \in C$.

5 Methodology

This chapter describes the methodology we propose to solve the problem of explaining Siamese Networks outcomes for audio inputs. Section 5.1 details the framework employed to approach *C-way one-shot* learning (introduced by Koch, Zemel, Salakhutdinov, et al. 2015), while Section 5.2 describes the novel method we propose to explain the outcomes in such a learning context.

5.1 C-way one-shot Learning

Similarly to the process followed by Koch, Zemel, Salakhutdinov, et al. 2015 and Honka 2019, we approached *one-shot* learning in two main phases: first we ***train*** the Siamese Network to learn pairs similarity on *training classes*, while ***validating*** it by monitoring the mean accuracy of different random C-way one-shot tasks on *validating classes*. Then, we ***test*** the network performance in terms of mean accuracy of several random C-way one-shot tasks on *test classes*. The learning-validating process is repeated until model convergence is reached, or no accuracy improvements are observed for a certain amount of validating iterations. Training, validation and test sets are disjoint, so to measure the model ability to generalise on unseen class distributions. Before going into the details of each step, let us illustrate a complete overview of this approach with Algorithm 1.

Algorithm 1 Overview of the general framework

```
1: repeat
2:     X, y ← GetTrainBatch()
3:     m ← TrainOnBatch(m, X, y)
4:     if current run involves evaluation then
5:         accuracy ← EvaluateOneShot(m, validation classes)
6:         save model weights if accuracy has improved
7: until model convergence or no validation accuracy improvement recorded
8: EvaluateOneShot(m, test classes)
```

The training-validating step is included in a loop cycle (lines 1-7, Algo 1) and the training stops if the model converge or if no validation accuracy improvement has been recorded in a given number of such loop iteration runs. The very first step is the training phase, where the model is trained on learning semantic similarities and dissimilarities between input pairs (line 3, Algo.1). What we refer to as m in this algorithm must not be confused with the f function defined in Chapter 4. Line 3 has to be intended as a preliminary phase where the model gets fit to training instances, to then being queried on a query sample and a support set to predict the class scoring the highest similarity value based on the previous training phase. The f function is encapsulated in the *EvaluateOneShot* function (line 5 and 8, Algo.1), since it is the one responsible to evaluate the model performance predicting new class instances similarities in the context of *C-way one-shot* learning. Training pairs are generated from the *GetTrainBatch* function (line 3, Algo.1), which is detailed as follows.

Algorithm 2 GetTrainBatch(*pos*, *ratio*)

```
Input: pos - positive pairs to generate,
       ratio - positive to negative ratio
Output: X - spectrogram pairs,
        y - spectrogram pairs labels

 1: X, y ← Inizialize()
 2: neg ← compute number of negative airs to generate from pos and ratio
 3: for class in training classes do
 4:     spect ← get a random sample from current class
 5:     for n in pos do
 6:         pos_spect ← get a different random sample from current class
 7:         X ← add pair <spect, pos_spect>
 8:         y ← add label 1
 9:     different_classes ← get neg classes different from the current one
10:     for diff_class in different_classes do
11:         neg_spect ← get a random sample from diff_class
12:         X ← add pair <spect, neg_spect>
13:         y ← add label 0
14: return X, y.
```

The batches used to train the models are formed of *positive* and *negative pairs*, labelled as 1 and 0 respectively. This labelling system guides the model in understanding when two inputs are somehow similar or dissimilar. The total number of

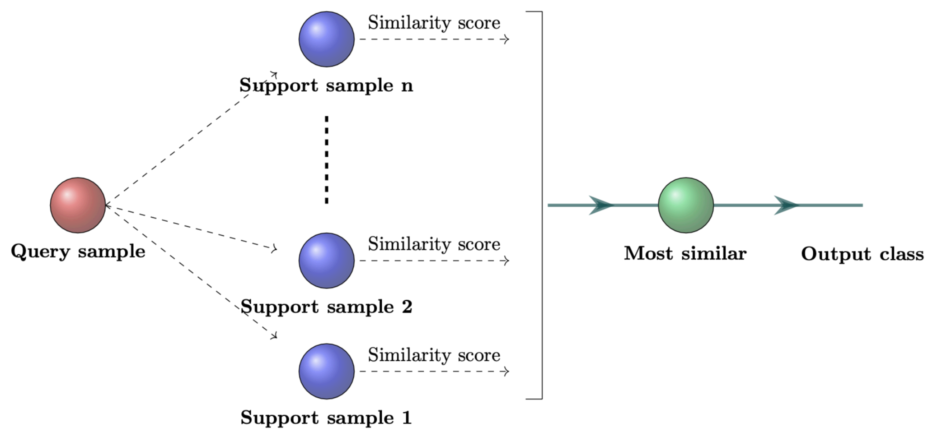


Figure 5.1 *One-shot classification process for C classes (Exhibited by Honka 2019).*

positive pairs is controlled via the ***pos*** parameter, while its combination with ***ratio*** defines the overall number of negative pairs to be generated (line 2, Algo. 2). Let us say we want to generate 4 positive and 2 negative pairs per batch, then we would set $pos = 4$ and $ratio = 2 \div 1$. Such parameters setting enables to study how data balancedness influences the model performances. Each training batch results in $C \times neg$ negative and $C \times pos$ positive pairs, where C is the training set class cardinality. For each class a first sample is picked at random and it is paired with pos different spectrogram chosen in a randomise way (lines 4-8, Algo.2). A number of neg samples are then selected randomically from the remaining classes, and paired with spectrogram picked at the very beginning (line 12, Algo.2).

While training, we regularly evaluate the model performance by measuring the mean *C-way one-shot* accuracy on C validation classes. We do this by generating several random *C-way one-shot* classification tasks, where each consists of comparing a ***query sample*** from class $c_i \in C$ to each element $s_i \in S$ of the ***support set*** as shown in Figure 5.1. The support set is composed of C samples, each belonging to a different validation class. Only one element of the support set belongs to the same c_i class that the query sample belongs to. A pair-wise similarity score is computed between the query sample and each element of the support set. Finally, the classification outcome is determined according to the support set sample scoring the highest similarity prediction. This process is repeated for every $c_i \in C$, and the mean accuracy of a given class c_i is computed considering several different iteration where class c_i is fixed as *query class.* Let us discuss the evaluating process detailed in Algorithm 3.

Algorithm 3 EvaluateOneShot(*model*, *runs*)

```
Input: model - model to evaluate,
       runs - evaluation runs per class
Output: accuracy - normalized global mean accuracy

1: accuracy ← 0
2: for class in evaluating classes do
3:     for run in runs do
4:         X, y ← GetOneShotBatch(class, evaluating classes)
5:         probs ← predict model probabilities on X
6:         if argmax(probs) == argmax(y) then
7:             accuracy ← accuracy + 1
8: return accuracy / (evaluating classes cardinality × runs).
```

We are interested in monitoring the overall mean accuracy considering all C classes by looping on a specified ***runs*** number of evaluation runs per each of these classes (lines 2-3, Algo.3). In each run we generate a C-way one-shot classification task, and increment the global accuracy when the highest similarity value matches the only pair labelled as 1 (lines 4-7, Algo.3). Finally, we normalize the overall accuracy in the range of *[0,1]*. It is important to draw attention to the fact that in Algorithm 3 we use the term *evaluating classes*, which could indicate classes deriving from both *validating* or *test* sets separately. The *GetOneShotBatch* function (line 4, Algo.3) is responsible to generate one-shot pairs batch on which the model is evaluated on. For the sake of completeness, let us show its specifics in Algorithm 4.

Algorithm 4 GetOneShotBatch(c, *evaluation_classes*)

```
Input: c - query sample class,
       evaluation_classes - validation or test classes
Output: X - spectrogram pairs,
        y - spectrogram pairs labels

 1: X, y ← Inizialize()
 2: query_spect ← get a random sample from query class c
 3: pos_spect ← get a different random sample from query class c
 4: X ← add pair <query_spect, pos_spect>
 5: y ← add label 1
 6: for every other class diff_class in evaluation_classes do
 7:     neg_spect ← get a random sample from diff_class
 8:     X ← add pair <query_spect, neg_spect>
 9:     y ← add label 0
10: return X, y.
```

Since the model expects two inputs, we have to generate a stack of C input pairs, where C is the overall number of evaluation classes. We do this by fixing the query sample class ***c*** and we pick at random the element which we refer to as ***query sample***. Then, another random sample from class c is selected, paired with the query sample and marked with label $y = 1$ to form a *positive pair* (line 4-5, Algo.4). Then, $C - 1$ *negative pairs* are formed: the query sample gets paired with $C - 1$ elements each belonging to a different class and labelled as 0.

Looking back at the general workflow described in Algorithm 1, we enter the final *test phase* once the model converges or stops training. This last testing phase is framed within the same framework used to evaluate the model while training, but on a separate test set. The final model performance is therefore measured in terms of mean accuracy of different *C-way one-shot* task using the process we previously discussed (Algorithm 3). The only difference is that here we make use of the test set, instead of the validating one.

This 2-tier ***train-validation*** and ***test*** framework allows to build the function f defined in Chapter 4. Evaluating the system in a *C-way one-shot* learning context in two separate moments and on two separate sets of class, is the core aspect of the methodology we followed. At the end of this process (Algorithm 3) we possess a

Siamese Network that, when presented with a query sample and a list of possible candidate - all belonging to unseen classes - it returns the one believed to be the most similar to the query input.

5.2 Explanation Method

The implementation of Siamese Networks we make use of comprehends two main parts: two identical convolution-based encoding sub-networks and the external final layers that compute the difference between the encoded vectors to generate their similarity score. To analyze the network outcomes, we propose to consider all layers contributing to the final outcome. Employing a *gradient-based* explanation technique on the isolated sub-networks would highlight how they work when reducing the inputs in different convolution stages, but they would not consider the final layers and the way they affect the similarity score prediction. Grad-CAM (Selvaraju et al. 2019) technique has been tested replacing ReLU activation functions with leaky ReLU ones to avoid vanishing gradient problems. Poor overall explanatory results were observed, since only the input feature that would mostly stimulate the encoders were highlighted. For these reasons, we decided to follow a *perturbation-based* approach measuring the outcome similarity prediction after different input perturbations.

A brief description of the way inputs are processed is now useful to understand the context where our explanation method sets the ground. Audio inputs are treated via their *log-mel spectrogram* representation, resulting in matrix of size $x \times y$ (Figure 5.2). The x axes represents ***time***, while the y axes represents ***frequencies***. Each *(time, frequency)* coordinate contains the decibel (dB) value of the recorded sample. Resulting x and y dimensions depend on the length of the recorder sample together with the parameter setting used when converting audios in spectrograms. Due to this-matrix structure a spectrogram representation could be intended as an image, but it should not be confused as one. In fact, it is crucial to add an additional *channel dimension* so that the spectrogram can be used as image-like input data with common convolution-based architectures.

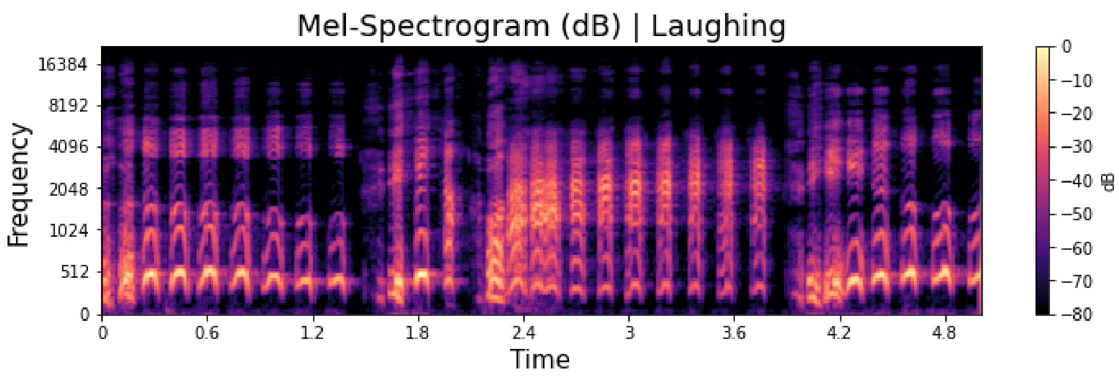


Figure 5.2 *Log-mel dB spectrogram of an audio containing a person laughing. Darker areas represents lower dB, suggesting silence in that frequency areas, while lighter pixels represent sounds that are audible by human hear.*

Before exploring the proposed explanation method, let us separate the general workflow to compute features contribution values in the following two steps:

1. Decide the segmentation technique;
2. Determine the input(s) to perturb and perturb them to store segments contribution values.

5.2.1 Segmentation technique

The first aspect to considered is how to perturb spectrogram inputs. After a first inspection of the classical *Occlusion* inspired scrolling window (Zeiler and Fergus 2013) and an analysis of the 3 audio segmentation technique proposed by Mishra, Sturm, and Dixon 2017 described in Chapter 2, we decided to test something different. The main limitation of occluding with a fixed size shape or segmenting on time-frequency areas, is that the morphology of the spectrum is not considered and therefore yield only approximate results. Occlusion experiments with scrolling window of shapes 1×1, 3×3, 50×50 and 100×100 were not able to find areas that would really affect the outcomes. Not quadratic-shaped window of different sizes were also tested, but the areas presumed to be important turned out not to be so when the model would be later queried on them as counter proof. Even worst, using fixed length segments to divide the input based on the time or the frequency axes individually, would imply a direct bond between specific class instances and their segmentation. Relying deeply on time segmentation alone, would mean assuming that every sound event starts and end at the same moment in time in a given recording. It is easy to see that in real-world applications it is unimaginable to assume that the actual sound event occur in the exact same time under different recording repetitions. In fact, this is hardly true even for recordings belonging to the same class. On the other hand, it is hard to segment a spectrum into separate but still meaningful parts. It is reasonable to believe that similar instances involve activations in comparable frequency bins, regardless of the recording repetition and the decibel intensity. But still, a slight decibel variance could be discriminative of two distinct classes and we want our model to be able to catch on that.

Considering the image-like matrix structure of spectrogram, we propose to segment them by means of techniques typically used for image inputs. The first algorithm we propose to use is *Felzenszwalb* introduced by Felzenszwalb and Huttenlocher 2004, which computes the Felsenszwalb's graph based image segmentation using a minimum spanning tree based clustering on the image grid. The second algorithm we propose to use is *SLIC* introduced by Achanta et al. 2012, which instead segments images using k-means clustering. Unfortunately the single-fake-channeled nature of audio inputs, precludes the possibility of experimenting with various other segmentation algorithms which process mainly *RGB* images. For this reason *Felzenszwalb* and *SLIC* are the algorithm we decide to cover, since they can work on grey-scale image-input as well.

5.2.2 Perturbation procedure

Once the segmentation technique is determined, the only thing left to decide are the input to segment and the actual procedure to later perturb it. But, considering the *C-way one-shot* framework, we have to think of a way to have an explanation that considers both the query and the support samples and compares them as evenly as possible. One could think of segmenting the query sample spectrogram and then measure the perturbation changes of its segments when overlayed on the support sample inputs. This approach would consider the *query sample* spectrum morphology and use that to compare evenly all the support set instances, which are different from the query by definition. A different approach would instead imply to segment each support set sample, and later perturb it on its own segments. Differently from the first procedure, here every support set sample gets segmented and perturbed based on its own characteristics.

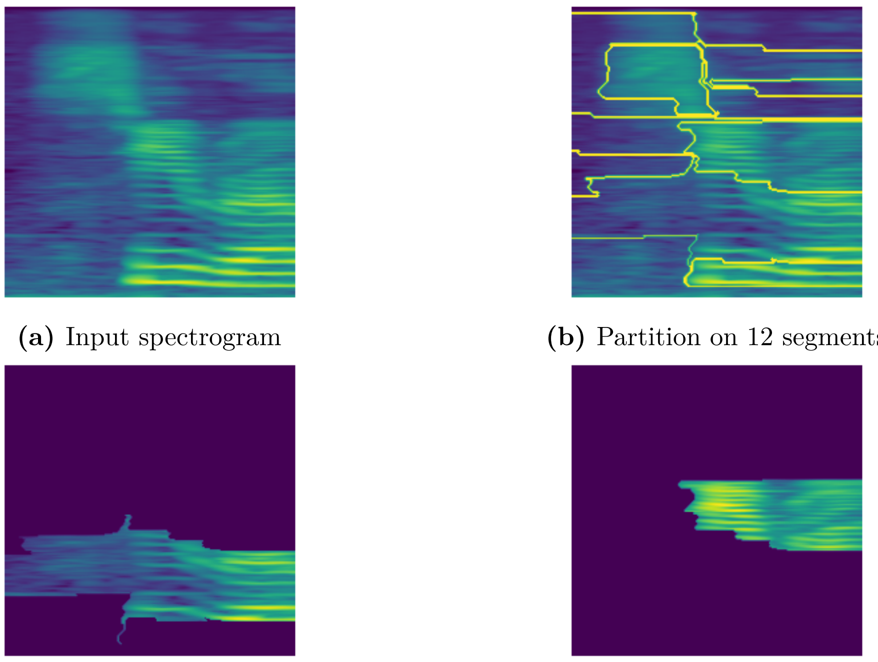
The overall idea we follow to explain the network outcomes in our *C-way one-shot* environment is outlined in Algorithm 5.

Algorithm 5 GetContributions(X, *model*, *seg_algorithm*)

```
    Input: X - one shot pairs batch,
           model - model to explain
           seg_algorithm - segmentation algorithm
    Output: contributions - contribution values

 1: contributions ← Inizialize()
 2: query_sample ← get query sample from X
 3: for supp_sample in the support set of X do
 4:     start_similarity ← predict model on <query_sample, supp_sample>
 5:     segments ← segment supp_sample with seg_algorithm
 6:     supp_sample_contribution ← Inizialize()
 7:     for segment in segments do
 8:         pert_sample ← perturb supp_sample current segment
 9:         pert_similarity ← predict model on <query_sample, pert_sample>
10:         delta ← start_similarity - pert_similarity
11:         segment_contribution ← delta / current segment size
12:         supp_sample_contribution ← update supp_sample segment_contribution
13:     contributions ← add supp_sample_contribution
14: return contributions.
```

The proposed method keeps the query input fixed (line 2, Alg.5) and, for every sample of the support set, it measures the model similarity outcome between the query input and the support sample before perturbation (line 4, Alg.5). Then, our method segments the support set sample using the selected segmentation algorithm (line 5, Alg.5). For each segment its contribution value is computed as follows: first the support set sample gets perturbed (line 8, Alg.5), then the new similarity outcome is computed pairing the query sample with the perturbed version of the support set spectrogram (line 9, Alg.5) and the difference between the starting


(a) Input spectrogram **(b)** Partition on 12 segments

(c) Perturbation on the 3rd segment **(d)** Perturbation on the 5th segment

Figure 5.3 *Overview of the perturbation procedure.*

similarity score and the one resulting after-perturbation is finally computed (line 10, Alg.5). Such difference is then weighted by the current segment size (line 11, Alg.5). When perturbing a spectrum based on a segment (line 8, Alg.5), we keep turned on that segment area while silencing off the remaining ones. Considering that we are dealing with audio inputs, *turning off* a segment means setting its value to -80 being this the smallest value in the dB scale. We could think this as a way to create silence in the area we want to turn off. Figure 5.3 shows an overview of this perturbation procedure: given a starting spectrogram (Fig. 5.3a), it is divided in segments by the segmentation algorithm (Fig. 5.3b) and then perturbed one segment at a time (Fig. 5.3c, Fig. 5.3d).

Before detailing the final version of the explanation method we propose to use, we want to draw the attention to some known problems *perturbation-based* methods usually lead to. First of all, when we perturb samples we might be generating out-of-distribution (OOD) data point. We have to remember that traditional models might generalize correctly on new samples, as long as they belong to a known distribution. One possible solution to this problem is to re-train the model on a dataset that includes the perturbed data points (Hooker et al. 2018), accepting the additional time resources needed. Ras, Çallı, and Gerven 2022 introduced the concept of *Hermitry Ratio* to indicate how close a data sample is to a given distribution and assessed that both the dataset and the architecture affect to some degree how perturbation-based methods generate OOD data points. On top of that we have to consider the *isolated effect* such techniques might lead to. Measuring the prediction changes of singular segments perturbations might help us understand how that segment is contributing to the final outcome, but it will disregard it completely from the interplay it has with the remaining input areas. In our work we considered both these problems and therefore included some fixes. Considering our few-shot learning context and the fact that we evaluate the model on new classes, we could think that it would not be affected by inputs that don't belong to the training set distribution. Despite this benefit, the nature of audio inputs might still play an important role when a spectrogram from an unknown distribution is compared with a perturbed sample belonging to that same distribution.

Our proposal to mitigate these downsides is to consider how much a specific segment contributes to the final outcome by considering its *average prediction* value. Inspired by *SHAP* main idea (Lundberg and Lee 2017), we want such value to consider the interplay between the segment and the remaining super-pixels. Differently from *SHAP*, our context does not allow to compute such value using the training set data. The goal of our project is in fact to produce a model-agnostic method which is disregarded from its training phase. We therefore suppose to have a trained model, and we want to explain its outcome when it is queried on new sets of classes. Let us now exhibit the refined proposed method in Algorithm 6 to better explain how we compute such *per-segment average prediction value.*

The improvements introduced to the first method described in Algorithm 5, are highlighted in yellow to help the reader spot the proposed mitigation fixes. To measure the interplay between each segment and the remaining areas of the spectrogram, we introduce a new parameter $\boldsymbol{p}$ that generates some additional perturbations where the segment in exam is turned on alongside different others. The overall number of segments that shall remain active in each perturbation is controlled via the $\boldsymbol{s}$ parameter. By doing so we measure not only how a specific segment influences the

Algorithm 6 GetContributionsRefined(X, *model*, *seg_algorithm*, p, s)

```
Input: X - one shot pairs batch,
       model - model to explain
       seg_algorithm - segmentation algorithm
       p - per-segment additional perturbations,
       s - per-perturbation active segments
Output: contributions - contribution values

1: contributions ← Inizialize()
2: query_sample ← get query sample from X
3: for supp_sample in the support set of X do
4:     start_similarity ← predict model on <query_sample, supp_sample>
5:     segments ← segment supp_sample with seg_algorithm
6:     supp_sample_contribution ← Inizialize()
7:     for segment in segments do
8:         segment_similarities ← Inizialize()
9:         perturbations ← compute p perturbations with s active segment each
10:        for perturbation in perturbations do
11:            pert_sample ← perturb supp_sample based on current perturbation
12:            pert_similarity ← predict model on <query_sample, pert_sample>
13:            segment_similarities ← add pert_similarity
14:        delta ← start_similarity - mean value of segment_similarities
15:        segment_contribution ← delta / current segment size
16:        supp_sample_contribution ← update supp_sample segment_contribution
17:    contributions ← add supp_sample_contribution
18: return contributions.
```

outcome, but how its interplay with other area of the spectrogram impact on the final outcome. For each additional perturbation, we store the similarity score obtained querying the model (line 10-13, Algo.6), and we compute the final segment contribution value as the difference between the starting similarity score and the mean similarity value obtained on such $\boldsymbol{p}$ additional perturbations (line 14, Algo.6). Finally, such *segment-average prediction* value is weighted according to the segment size (line 15, Algo.6). Similarly to what we previously described, this process is iterated over each pair of a *C-way one-shot* batch keeping fixed the query sample and applying such perturbation methodology to each element of the support set.

At the end of Algorithm 6, we hold every support set sample contribution values per each segment within its own spectrogram segmentation. Each support set contribution map has to be intended as an heatmap having the same $x \times y$ size of the sepctrogram input itself. We can therefore visualize such heatmaps showing how each segment is influencing - either positively or negatively - the final similarity score outcome. For visualization purposes, the contribution values were considered in terms of their absolute values and normalized in the scale of *[-abs(contributions), abs(contributions)]*. Finally, to target the problem of explaining each *C-way one-shot* classification task as a whole, we propose to share a common representation scale. We therefore use the negative($-$) maximum absolute value as minimum of such scale, and it positive($+$) variant as maximum of the scale.

6 Experiments

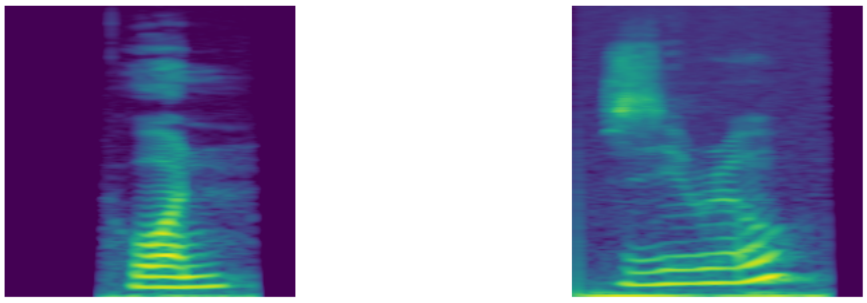
In this chapter we describe the different experiments we conducted to build and explain the architecture of interest. First of all, we introduce the datasets and detail how we pre-processed the data in Section 6.1, and we describe the network architectures in Section 6.2. We continue detailing how we trained, evaluated and tested the models in Section 6.3, concluding in Section 6.4 reporting and discussing qualitative examples of the explanation method we proposed.

6.1 Datasets

One of the core elements of our project is the audio nature of the input data. We utilize the *AudioMNIST*[1] and *ESC-50*[2] datasets, exploiting the *LibROSA* (McFee et al. 2015) package[3] to pre-process them.

The ***AudioMNIST*** dataset is composed of 30000 audio recordings of spoken digits (0-9) in the English language. Each digit is repeated 50 times for each of the 60 different speakers. These audios were recorded in quiet offices as mono channel signal at a sampling frequency of 48kHz using a RØDE NT-USB microphone. 12 of the speakers that recorded the clips were female, while 48 of them were male and their ages range between 22 and 61 years old. Considering the context of *C-way one-shot* learning we decided to use this dataset to pursue a *speaker recognition* task, due to the high number of speakers and digits repetitions. The high number of available speakers allows us to create three disjoint set: the training set is composed of *50* classes, while the remaining *10* are divided equally between validation and test set. The large number of available repetition for each of speakers, enables us to generate a reasonable amount of both training, validating and testing pairs. To transform the audio inputs, we decided to use their *logarithmic-mel spectrogram*

[1]`https://github.com/soerenab/AudioMNIST`
[2]`https://github.com/karolpiczak/ESC-50`
[3]`https://librosa.org/`



(a) A man saying one **(b)** A woman saying zero

Figure 6.1 *Examples of the AudioMNIST dataset spectrograms.*

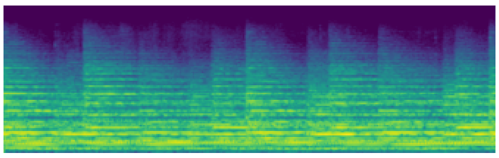
representation, which is commonly used for audio classification tasks (Hershey et al. 2016, Piczak 2015) and conveniently enough it allows to use architectures originally designed for image classification. Prior to the transformations, the original samples were down-sampled to 41kHz and zero-padded in order to generate input vectors equal in length. While zero-padding, we added silence to the audio (-80 dB) by placing the audio signal at random position within the vector. We followed this approach to add randomness to the data, preventing silence to be placed at the same position for all samples of our dataset. For each sound clip we computed its *log-mel spectrogram* representation using an FFT window size of 4096, hop length of 216 samples and 224 mel-bands[4]. Such process lead to spectrogram of sizes equal to 224×224, that would store enough information to allow an inverse transformation that helped us sonify the spectrograms in the later explaining phase. Figure 6.1 shows two examples of the resulting spectrograms for this dataset.

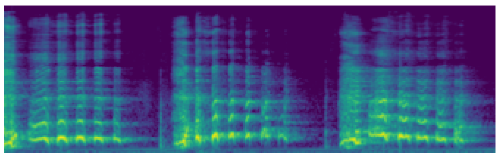
The **ESC-50** dataset is a collection of 2000 annotated 5-second audio clips divided into 50 different classes and 40 repetition per class. All dataset clips were recorded at a sampling rate of 44.1kHz. Similarly to the *AudioMNIST* dataset, the large category space of *ESC-50* is beneficial for Siamese Networks: a large amount of pairs can be created to train the model and then generalize on new unseen classes. In this case, we decided to pursue an *environmental audio classification* task. We split the dataset so to have *50* training class, *5* validating and *5* testing ones. The dataset recordings can be grouped into 5 macro-categories, each containing 10 classes. This division is shown in Table 6.1 to give the reader a general overview of the kind of

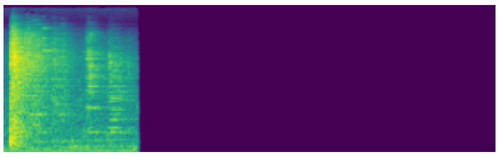
[4]Parameters specifics at `https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html`

Animals	***Natural soundscapes***	***Human, non-speech sounds***	***Domestic sounds***	***Urban Noises***
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	See waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

Table 6.1 *ESC-50 classes*


(a) Curch bells class spectrogram


(b) Laughing class spectrogram


(c) Glass breaking class spectrogram

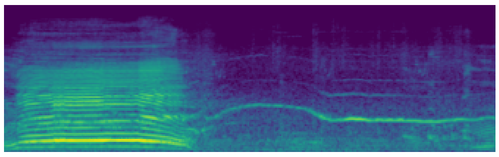

(d) Cow class spectrogram

Figure 6.2 *Examples of the ESC-50 dataset spectrograms.*

classes we will deal with. Here again, we processed the audio clips to recover their *log-mel* spectrogram representations. Each sound clip was converted using a FFT window size of 2048, hop length of 512 samples and 128 mel-bands. The dimensions of the each spectrogram converted on such parameters result to be of 128×431. In this occasion, the sampling rate was kept at 44.1kHz and no down-sampling was applied before conversion. Example of *ESC-50* spectrogrms resulting from this conversion process is showed in Figure 6.2.

Let us now discuss a final change we made to our data by quickly recalling that in our $x \times y$ spectrogram size definition, the x axes represents ***time*** while the y axes represents the ***frequency*** variable. Despite the fact that spectrograms might be intended as images due to their matrix structure, they still could not be programmatically used as such. An additional *channel dimension* was infact needed to use them as image-line input for common convolution-based architecture. The final spectrograms input dimension would therefore be $x \times y \times 1$.

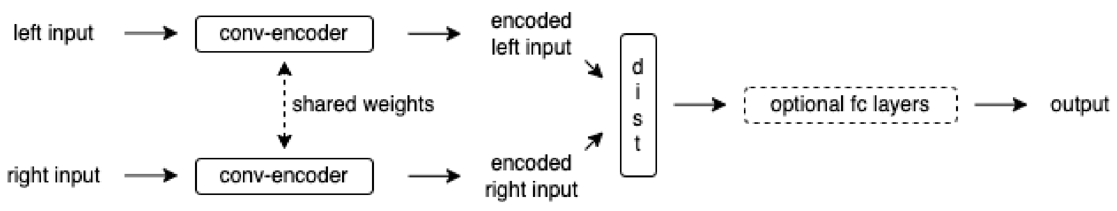


Figure 6.3 *Siamese Network overview*

6.2 Network architectures

In this project we built two distinct networks, one per each of the two dataset described. Before getting into their details, let us briefly present an overview of a typical Siamese Network that process spectrograms by means of Figure 6.3. The overall architecture is composed of two convolution-based encoders, followed by a distance (or merging) layer and a final output scoring layer. Some optional fully connected layers might be present right after the encoded inputs get merged together. The two encoders share the same architecture and weights, which are updated simultaneously during training so to result in identical embedding sub-net. Each conv-encoder is composed of several convolution blocks, usually followed by a flattening and a fully connected layer. The merging layer calculates element-wise metric distance between the embedded inputs. Finally, an output layer consisting of a single fully connected unit functions on such difference and generates a similarity output score between the two given inputs.

Spectrogram representation enables to use architecture originally designed for image-input like AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and VGG Net (Simonyan and Zisserman 2014). This is why the encoders of our networks were implemented as convolution-based architecture. Starting from this well-known architectures and inspired by Becker et al. 2018, we conducted several experiments to adjust the inner encoder structure. In addition, we experimented on different parameters configuration to find the best architecture for each of the two dataset. The hyper-parameters we tested include arrangements for the inner-structure of the encoders as well as variables controlling the overall architecture learning process. Table 6.2 lists all the parameters we investigated together with their search space.

Hyper-parameter	**Search space**
Number of conv layers	*1, 2, 3, 4*
Number of filters per layer	*12, 32, 64*
Pooling layers type	*Average Pooling, Max Pooling*
Fully connected unit size	*1024, 2048, 4096*
Distance metric	*Absolute difference, Squared difference*
Loss function	*Binary crossentropy, MSE, Contrastive loss*
Dropout in conv layers	*0.0, 0.25, 0.5*
Dropout in fully connected layers	*0.0, 0.25, 0.5*
Number of positive pairs	*2, 4, 6*
Positive-to-negative pair ratio	$1 \div 1$, $1 \div 2$, $1 \div 3$, $2 \div 1$, $3 \div 1$

Table 6.2 *Hyper-parameter configurations*

The hyper-parameter tuning phase was conducted in the learning-evaluating framework described in Chapter 5 and the mean *5-way 1-shot* accuracy was marked as the value to maximize. After some preliminary trial runs, the number of maximum training epochs was set to 1500. Such number resulted to be a reasonable trade-off between the model performance and the run time. After 50 training epochs, the model would be evaluated on 100 random *5-way 1-shot* tasks, and the training would stop if no mean accuracy improvements was recorded after 10 of such evaluation runs. The *Adam* optimizer (Kingma and Ba 2014) was used for the loss function minimization process. This framework was applied separately on the two dataset.

While tuning, we noticed some similarities. First of all, both dataset architecture would stop training due to the evaluation threshold when the *MSE* and the *Contrastive loss* were employed. These loss function would not let the model train since its training accuracy score would only decrease progressively. Another common behaviour was found when dropout was in work: the mean one-shot accuracy would result equal to 1 (the maximum) after few evaluation runs and never unlock from this state. After investigating this condition, we noticed that the model would somehow explode and return a similarity score of 1 (the maximum) for every support-set pair. To summarize, dropout would led the model in classifying every pair as completely equal always and forever on both datasets. After noticing such similarities between the models behaviour on the two different datasets, we decided to test the best resulting architecture for *AudioMNIST* to *ESC-50*. Surprisingly enough, the

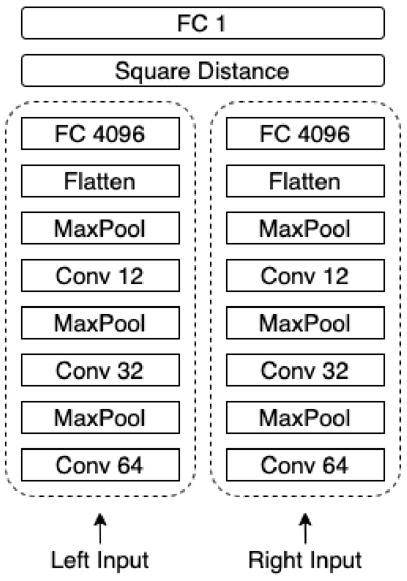
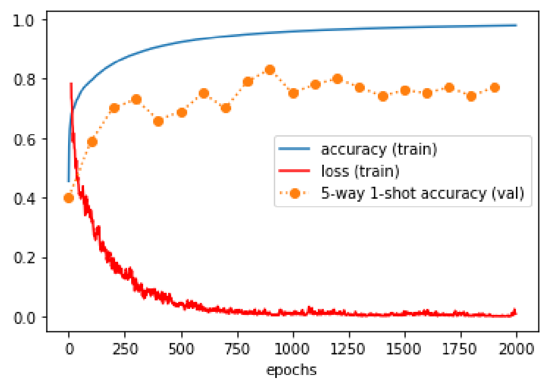


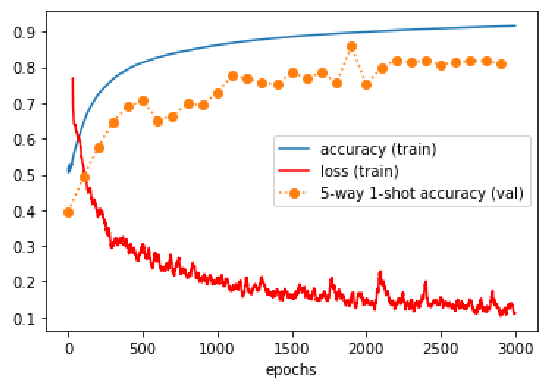
Figure 6.4 *Selected Siamese Network architecture.*

same architecture would achieve the best result on both the two dataset separately.

The final selected architecture, for both the two distinct dataset, is showed in Figure 6.4. The sub-encoders are composed of 3 convolution blocks, each containing a 2D-convolutional layer and a max-pooling layer. ***Conv 64*** is composed of 64 filters and a kernel window of size 5×5, ***Conv 32*** contains 32 filters and a 5×5 kernel size while the last ***Conv 12*** layer is composed of 12 filters with a 3×3 kernel window. Each convolution layer is activated with a rectified linear unit (ReLU) activation function. The ***max-pooling*** layers all share a 2×2 window pool size. The max-pooling layers down-sample the previous layer's output applying a max-filter to sub-regions of specific size, so to reduce the complexity of the input at different stages. Finally, a ***fully connected*** layer of 4096 units takes the input back to a 1-dimension form. The ***square distance*** operates as merging layer between the two encoded inputs, and a final fully connected layer composed of only 1 unit computes the similarity score by means of a sigmoid activation function. The ***positive-to-negative*** pair ratio was selected to be $1 \div 1$, since a slight tendency to overfit was noticed if it would tip to either one of the two pole. The only difference was in the number of ***positive (and negative) pairs*** to generate: *2* and *4* were selected for AudioMNIST and ESC-50 respectively. In both cases, a higher number would drastically slow the training procedure, without leading to better results.



(a) MNIST-Net



(b) ESC-Net

Figure 6.5 *Training accuracy (blue) and loss (red) curves together with the 5-way 1-shot mean accuracy progress (orange). The validating mean accuracy was measured once every 100 epochs.*

6.3 5-way one-shot learning performance

Let us quickly summarize was we previously described. Two different Siamese Networks were built, one per dataset. They were trained to learn pair similarities on a training set while being validated in a *C-way one-shot* learning framework on a validation set, and then tested on a test set. The training/validation/test set split results in ***50/5/5*** and ***40/5/5*** classes for the *AudioMNIST* and *ESC-50* dataset respectively. Let us now give the details of the training, validation and test phase that the final selected architectures have gone through. In our discussion, we will refer to the two networks as *MNIST-Net* and *ESC-Net*.

Differently from the hyper-parameter tuning phase, the maximum number of training epochs was increased to 5000, with the model performance being evaluated still every 100 training runs but this time on 300 random *5-way 1-shot* tasks. While training, the model weights were updated retrieving the layers gradients with respect to the ***binary crossentropy*** loss value after every training batch. Figure 6.5 shows the *training-validating* process for the two distinct dataset. Both the training procedure stopped because no *5-way 1-shot* mean accuracy improvements was recorded in 10 evaluation runs. *MNIST-Net* reached its best mean 5-way one-shot accuracy after only 900 training epochs with a value of ***0.83 (83%)***, while the process for the *ESC-50* dataset needed 1900 training epochs with a pick value of ***0.86 (86%)***. Additional tests with higher value of the early-stopping threshold were explored, but

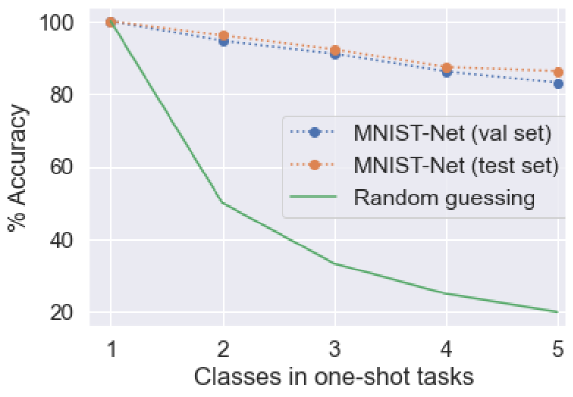
Class	Accuracy
04	93%
56	91%
55	88%
27	82%
46	78%
Mean total	86%

(a) *MNIST-Net*

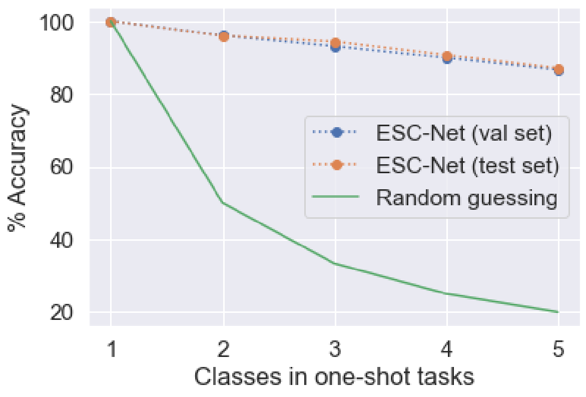
Class	Accuracy
Glass breaking	99%
Church bells	93%
Frog	91%
Laughing	82%
Door wood creaks	71%
Mean total	87%

(b) *ESC-Net*

Table 6.3 *5-way one-shot mean test accuracy scores.*



(a) MNIST-Net



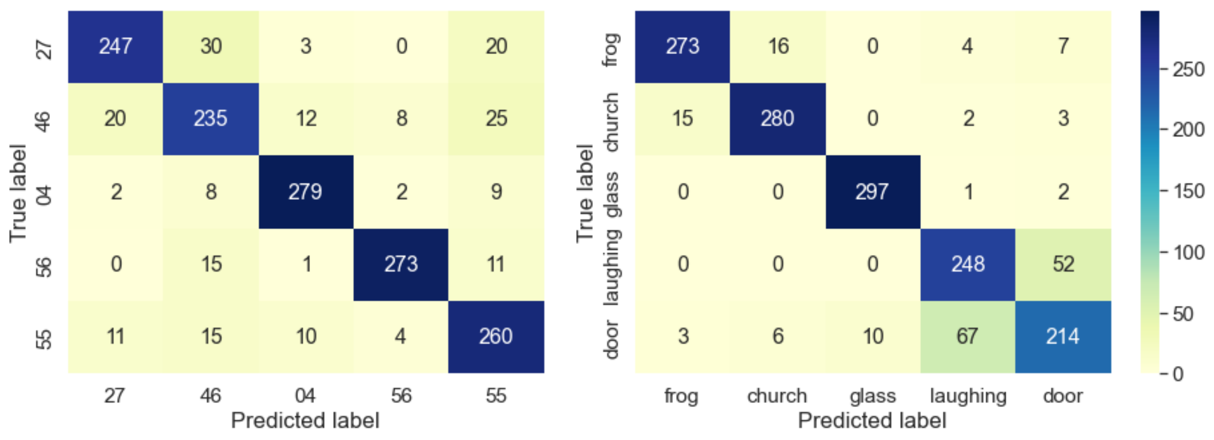
(b) ESC-Net

Figure 6.6 *C-way one-shot mean accuracy while varying C from 1 to 5.*

no overall improvement was found. The additional time needed by *ESC-Net* might be due to the fact that it is composed of a smaller number of training classes. Each training batch would therefore result in a lower overall number of pairs to learn from.

Both networks were tested on test sets composed of 5 unseen classes each[5]. To better comprehend the models behaviour, the mean *5-way one-shot* accuracy was also measured per each singular class. Test results for both networks are listed in Table 6.3. Results show that the network working on the *ESC-50* dataset reaches accuracy values higher than 90% on 3 out of 5 classes, but the *door wood creaks* class lowers the overall value scoring a mean accuracy of 71% singularly. Such value is smaller than the smallest *MNIST-Net* results in, which generally scores values

[5]Not even one sample of the test classes was ever seen during training, leading to a situation which is commonly referred as zero-shot learning. Despite this, in this thesis we use the term C-way one-shot learning to indicate that the additional support set is composed of a singular sample for each of the C classes. We are therefore asking the model to zero-shot the right classification of an unseen query class, by providing exactly one other sample of that same class in the support set.



(a) MNIST-Net **(b)** ESC-Net

Figure 6.7 *Confusion matrix classification results.*

that seems to be better distributed among the 5 classes. The final mean accuracy results to be of ***86%*** for *MNIST-Net* and ***87%*** for *ESC-Net*. A slight improvement of *1%* and *3%* has been recorded from the validation phase for the two networks respectively. This is not surprising, since we evaluated the model measuring the mean accuracy in a *5-way one-shot* framework. Figure 6.6 compares the model accuracy while varying C from 1 to 5 on both validation and test set. The mean accuracy decreases at increasing values of C, making it harder for the model to match the right support set element for a given query input. Interestingly enough, the mean accuracy decreasing behaviour is somewhat similar between the validating and test set on both networks. This might be a counter-proof to the fact that the model is interested in the samples semantic features, more than their class distribution. We can safely state that the training-while-validating framework has successfully trained the model to generalize on new classes.

Finally, let us now visualize the results achieved by the model in form of confusion matrix in Figure 6.7. Following on what the mean accuracy value results suggested, we can see that while the miss-classification are almost equally distributed in *MNIST-Net* (Fig. 6.7a), higher miss-classification values are found for *ESC-Net* (Fig. 6.7b) between the *laughing* and *door wood creaks* classes. This might be due to both *laughing* and *door wood creaks* consisting of sounds generally composed of repeated repetition distributed along a high range of the frequency spectrum.

6.4 Explainability

Let us recall that the goal of our explainability method is to explain *5-way 1-shot* classification tasks in their entirety, highlighting which segments have a positive influence and which others a negative one on the similarity outcome.

The two segmentation algorithms we tested are *Felzenszwalb* and *SLIC* via their publicly available implementation of the **scikit-image**[6] python package. *Felzenszwalb* implementation opens to test on different parameters like *scale* to indicate the cluster dimension scale, *sigma* to specify the standard deviation of the Gaussian kernel used while pre-processing, and *min_size* to suggests the minimum size of each segment. Regarding the *SLIC* algorithm, the package opens to a variety of parameters to test with and it even allows to use the *maskSLIC* version of the algorithm (Irving 2016) which is commonly used to segment medical images. Both algorithms detailed documentation can be found on the official package website[7]. After different preliminary test, we decided to use the *Felzenszwalb* segmentation algorithm on both dataset with the same parameters setting of $scale = 100$, $sigma = 1.5$ and $min\_size = 1000$. The number of random perturbation to be generated per each segment was selected to be *500* (parameter $\boldsymbol{p}$, Algorithm 6) and the percentage of segments to keep active along with the examined one in each perturbation was set to *20%* (parameter $\boldsymbol{s}$, Algorithm 6). Such configuration resulted to be a good trade-off between the number of segments per each sample, the explanatory results of the attribution values computed and the time needed to generate them. The visualization color scale of our explanations ranges from light blue to pink: blue areas represent a negative influence on the model outcome, while pink portions indicate a positive contribution to the similarity score. White areas are instead neutral to the model classification. Important segments were further analyzed and an inversion from spectrogram to audio was applied on such areas by means of the Griffin-Lim algorithm (Griffin and Lim 1984). We will refer to this process as *Soundification.* We carried out a total of 100 experiments per each dataset, dividing them in 20 per class. Each of the 20 class-related experiment would consider that specific class as the query label.

[6]`https://scikit-image.org/`

[7]`https://scikit-image.org/docs/stable/api/skimage.segmentation.html`

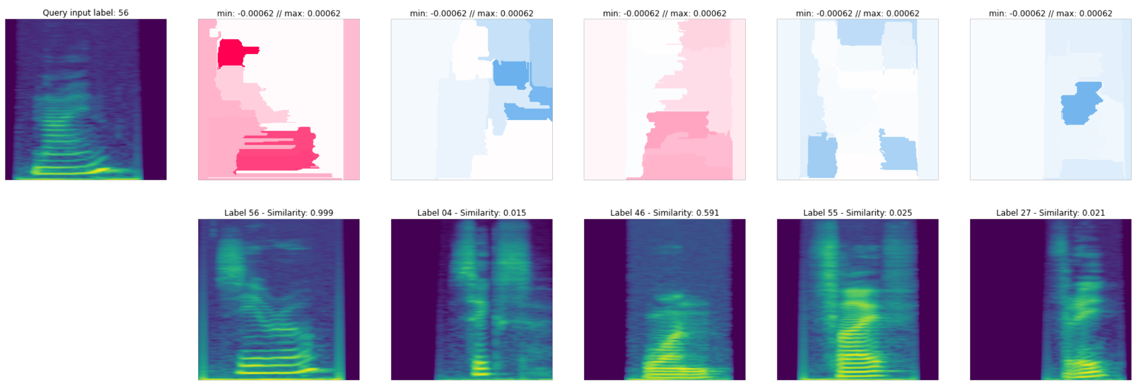


Figure 6.8 *Explanation of a 5-way one-shot correct classification on speaker 56. MNIST-Net outcomes a similarity score of 0.99 for the support sample labelled as 56, while the second most similar spectrogram achieve a similarity score of 0.59 and it was produced by speaker 46.*

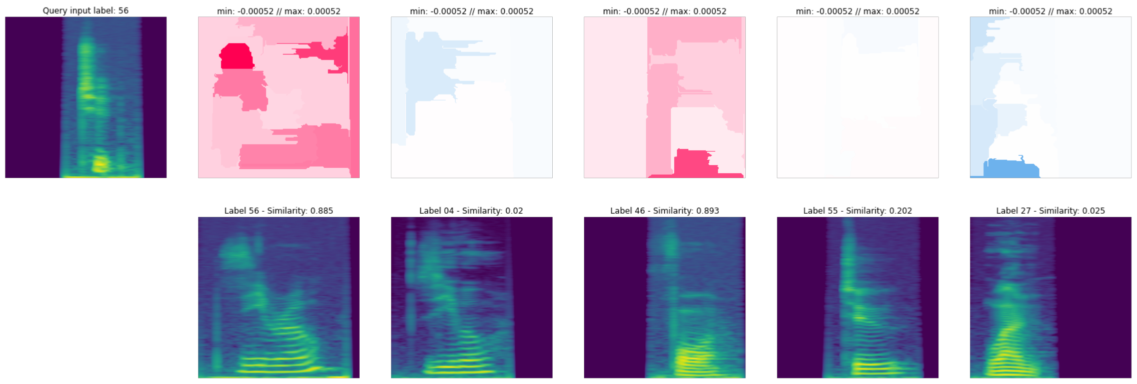


Figure 6.9 *Explanation of a 5-way one-shot incorrect classification on speaker 56. MNIST-Net outcome a similarity score of 0.885 for the support sample labelled as 56, while the most similar spectrogram achieve a similarity score of 0.893 and it was produced by speaker 46.*

6.4.1 AudioMNIST Results and Discussion

The test set used to generate the explanation for the *AudioMNIST* dataset is composed of the following speaker classes: 56, 27, 46, 55, 56. Speaker 56 is the only female, while the remaining ones are male. Several experiments were conducted for each of the speakers in order to understand the model outcomes on both correct and incorrect classifications.

Let us examine an example of correct classification for *speaker 56*, which is shown in Figure 6.8. In this example the model returns a similarity score of *0.99* for the right support set spectrogram. By inspecting its heatmap, it is easy to see that darker segments contain higher dB values, meaning that the model is indeed looking at spoken areas. To be more specific, the outcome is due to segments that range between ***200 and 1000Hz*** along with frequency areas ***higher than 4096Hz***. In a general prospective, this explanation brings to light what is important to the model and why the other support samples did not score high similarity values. Speakers labelled as 04, 55 and 27 reach similarity score lower than 0.1 because no areas rising such outcome value were found. Moreover, we can see that despite the fact that similarities are found between *speaker 46* and the query sample, they have a much lower magnitude impact than the ones found on the correct support sample.

By looking at the miss-classified example for the same *speaker 56* in Figure 6.9, an interesting result can be observed. The sample wrongly classified scores a similarity value of 0.89%, while the right spectrogram is only 0.01% smaller. By inspecting this miss-classification, we can clearly see that the frequency spectrum the model is looking at for wrong label 46 range from ***0 to 512Hz***. These values are much smaller than the ones considered to be important when label 56 gets correctly classified.

The classification confusion matrix in Figure 6.7a shows that the higher value of miss-classification for speaker 56 is indeed the speaker labelled as 46, and quantitative explanation experiments suggest what the model behaviour might be. When it correctly classifies *56 speaker*'s audio, it mainly considers spoken segments in medium-high values of the frequency spectrum. On the other hand when miss-classifying, it mainly looks at areas located in at the very bottom of the frequency range. A similar behaviour was also found when the query sample is spoken by any other male speaker. In Figure 6.10 we show an example of miss-classification for the male speaker 55 to comment on a common classification behaviour we noticed. In this specific case we can see that the support set sample scoring the highest value belongs to speaker 46 and that important segments are found in frequencies higher than 4096Hz. Such frequencies are completely different from the ones the model is looking at the majority of the time when correctly classifying audios of the query sample speaker 55. Quantitative experiments guide us to believe this is a mistake

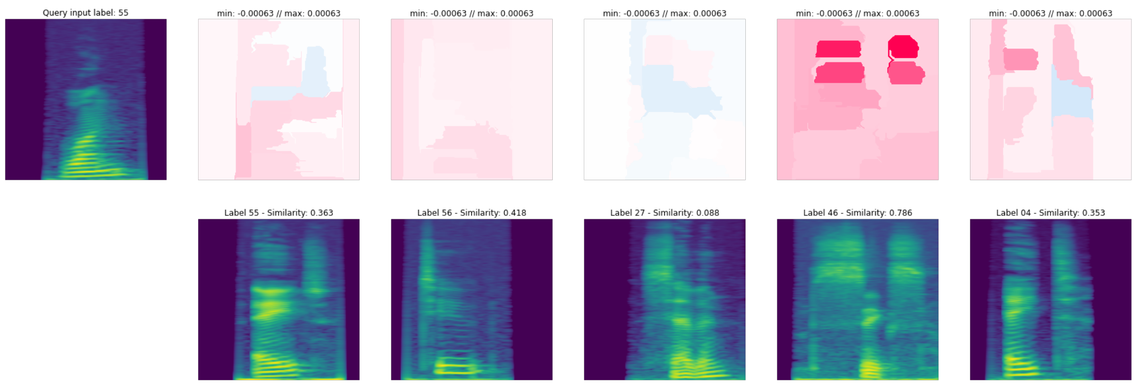


Figure 6.10 *Explanation of a 5-way one-shot incorrect classification on speaker 55. MNIST-Net outcomes a similarity score of 0.363 for the support sample labelled as 56, while the most similar spectrogram achieve a similarity score of 0.786 and it was produced by speaker 46.*

the model repeats frequently. A correct classification for a specific male class would usually look at lower freuency Hz values, while incorrect classifications would instead usually depend on segment higher in the frequency spectrum.

Let us now draw some conclusion for *MNIST-Net* from the experiments we carried out. First of all, each perturbation run on the described parameters setting would need an average time of *1 minute (*$\pm 0.5min$*)* per each element of the support set. In our case, considering the 5 classes composing the support set, the overall time needed to have a complete explanations on a *5-way 1-shot* task would be of *5 minutes (*$\pm 2.5mins$*)*. Experiments show that, generally, positive-influential segments are usually the same for a specific speaker, and they change according to the query speaker in exam. The female speaker's important areas would always reside in medium-high frequency areas (higher than 2048Hz), while the male speaker would all lay together in the lower spectrum (smaller than 1024Hz). Male speakers have also slight differences between them, but the model seems to catch on them correctly. Speaker 46 has the lower tone of voice (<200Hz), while speaker 27 speaks a little higher (150-300Hz). Finally, speakers 55 and 04 progressively speaks in the range from 250-350Hz and 300-450Hz respectively. In many occasion, the *MNIST-Net* tendency of focusing on silent spectrogram areas has been highlighted. Although experiments did not show cases where a correct classification is mainly due to silence portions, we could still examine miss-classification that predominantly depended on such areas from time to time. This wrong behaviour might be due to the data-augmentation phase we carried out inserting silence areas at random position to uniform the dataset. The soundification process did not help in drawing additional conclusions. Listening back to both positive and negative influencing segment is not very helpful in this context, since the overall recording duration is of 1 second. Segments would only contain portions much shorter and, therefore, with a poor meaning. Listening to male speakers important segments is also pointless since they all mostly lay in the same range of low-frequency. Such lower frequencies are hard to be perceived as different by human ear, especially when they do not include their harmonics and occur in very short sound events.

6.4.2 ESC-50 Results and Discussion

We tested *ESC-Net* performance on a test set composed of the following classes that we now seek to explain: *frog, curch bells, glass breaking, laughing* and *door wood creaks*. The classification confusion matrix depicted in Figure 6.7b shows that the first three classes seem to be recognizable enough, while the last two result in the highest miss-classification rate instead. We conducted several experiments on each class fixing it as query label, to try understand the model classification behaviour.

Let us first analyze the *laughing* and *door wood creaks* classes, which are frequently mixed-up together. Inspecting different *5-way 1-shot* tasks, we noticed that both classes provide samples belonging to a wide domain space: *door wood creaks* recordings would vary from short sounds to longer ones with wood singular creak being alternatively well separate or almost completely unified. Same thing goes for the *laughing* class: laughs might appear both as unified spectrograms or well-separate laugh giggles. Figure 6.11 shows an examples of correct classification for the *door wood creaks*. If we take a closer look to both query and support set samples belonging to this class, we can see a big difference by sight alone. The query sample is composed of similar creaks repetitions, each spanning across the entire frequency spectrum range, that tend to create a uniform sound as a whole. The support set spectrogram is composed of much fewer creaks instead.

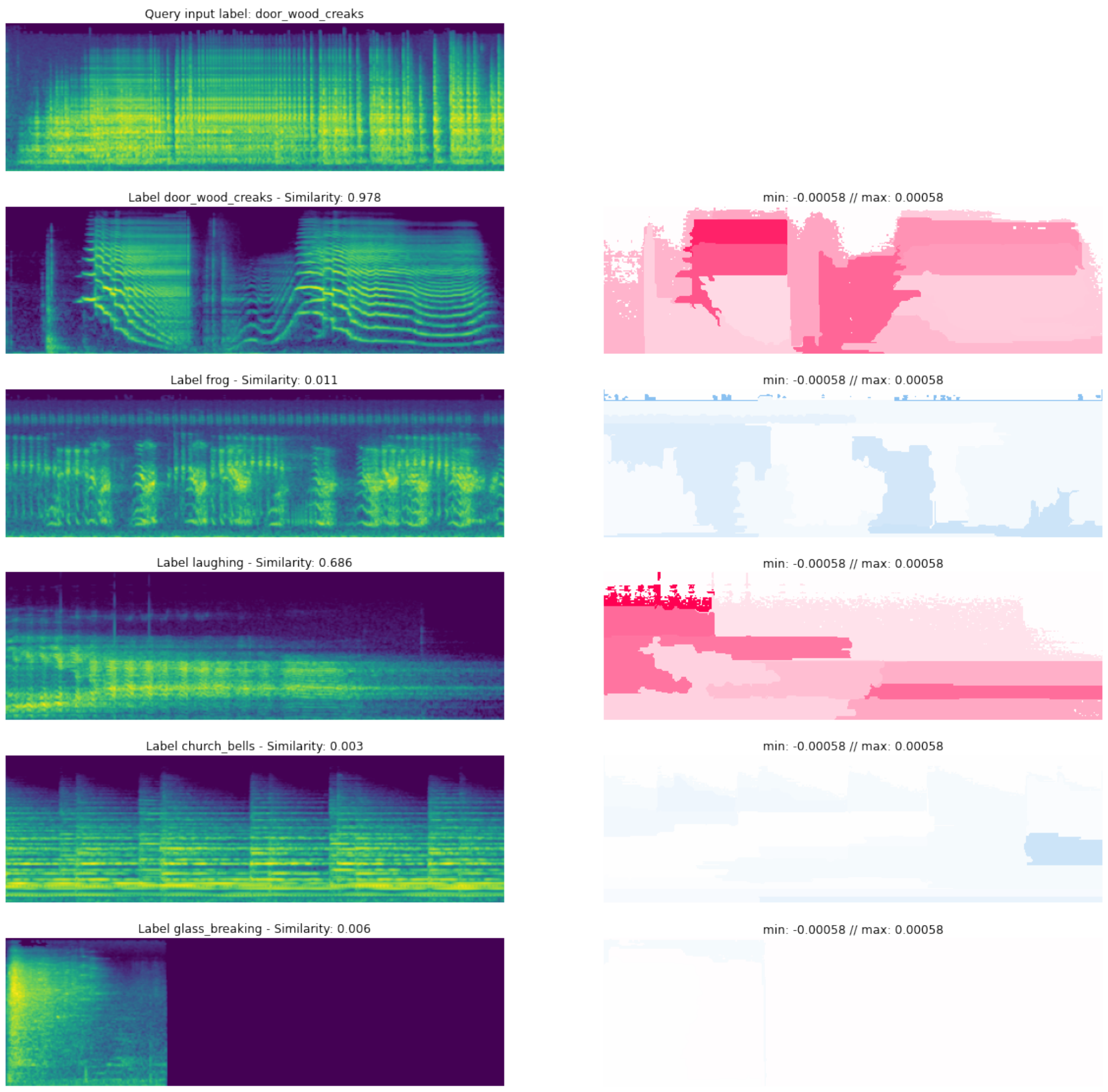


Figure 6.11 *Explanation of a 5-way one-shot correct classification on door wood creaks class. ESC-Net outcomes a similarity score of 0.978 for the support sample labelled as door wood creaks, while the second most similar sample achieve a similarity score of 0.686 and it belongs to the laughing class.*

Moreover the support sample spectrogram presents two main frequency slides: the first slides down frequencies from 4096Hz up to 0Hz, while the second slide does the exact opposite. Our explanation brings to light the fact that *ESC-Net* is mostly paying attention to the *door wood creaks* support samples portion that occur between 0.6 and 1.8 seconds. This segment is located at high frequencies (>4096Hz) and it is composed of a number of repetition creaks which is higher than it is in the rest of the spectrogram. This is interesting to notice because this segment area is very much similar to the query sample morphology itself. The explanation also

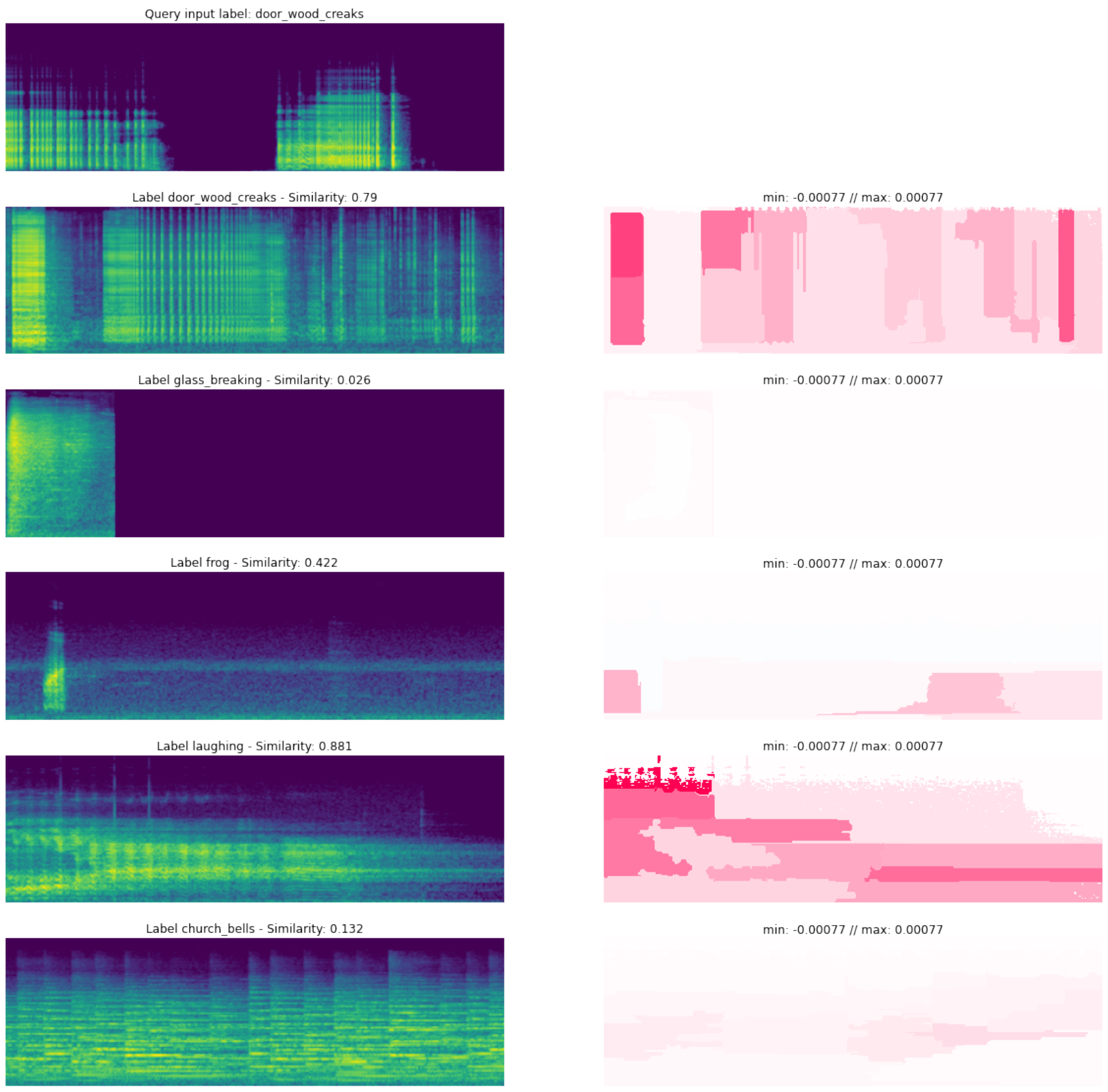


Figure 6.12 *Explanation of a 5-way one-shot incorrect classification on door wood creaks class. ESC-Net outcomes a similarity score of 0.79 for the support sample labelled as 56, while the most similar spectrogram achieve a similarity score of 0.881 and it belongs to the laughing class.*

highligts that the *laughing* support sample is considered somewhat similar to the query input due to the same higher frequency range, especially in the wave crests of each singular laugh. Through different experiments we could also notice that when *ESC-Net* correctly classifies the *door wood creaks* class samples, the remaining *frog, curch bells and glass breaking* classes are mostly only colored in blue.

Let us now exhibit an example of incorrect classification for the *door wood creaks* class in Figure 6.12. This example, together with the correct-classification one we previously described, were carefully chosen because they encapsulate the two main problems that different experiments showed us to affect the model in mixing-up these two classes rather frequently. The first of them being the wide domain range of both classes recordings, and the second one is the inability of discriminate correctly on the higher frequency range. Here again, if we take a look at the samples themselves we can see their different spectrogram morphology. The query *door wood creaks* sample in Figure 6.12 is composed of separate creak sounds that, differently from the support sample belonging to the same class, reside in a smaller range of frequencies (<2048Hz). Such condition is enough for *ESC-Net* to classify the query sample as *laughing*, since the wave crests of each laugh mostly lives in that same frequency range.

Regarding the *frog* and *church bells* recordings we could notice a dual attitude of *ESC-Net*. The network classifies *frog* samples correctly when the different croaks are well-separate, while it is prone to miss-classify them as *church bells* clips when the distance between croaks is relatively small and it includes a little decay after the sound event itself. In a dual manner *ESC-Net* correctly classifies *church bells* if the different bell tolls are somehow connected by means of delay, while it classifies them as *frog* recordings if they are well-separate. Finally, the *glass breaking* samples are pretty much different from all other classes since they are typically composed of short sound events (1,2 seconds) in a longer audio recordings (5 seconds). Usually for this class, our methods exposes as important areas for correct classifications a curve that follows the spectrogram separation between the end of the breaking-glass sound event and the remaining silence exclusively. The tendency to miss-classify the *glass breaking* recordings is especially highlighted when the query sample of a glass breaking is longer in time, the relative support sample is instead shorter and the *door wood creaks* class spectrogram has a length much more similar to the query input than the supposedly correct support sample does.

Let us now draw some conclusion for *ESC-Net* from the experiments we carried out. Each perturbation run on the described parameters setting would need an average time of *1.5 minute (*$\pm 1.0 min$*)* per each element of the support set. In our case, considering the 5 classes composing the support set, the overall time needed to have a complete explanations on a *5-way 1-shot* task would be of *7.5 minutes ($\pm 5 mins$).* A common mistake that *ESC-Net* does when miss-classifies some query samples is due to the spectrograms actual morphology regardless of the query class in exam. Our explanation method seems to indicate that the network is indeed able to discriminate between different classes as long as the query sample and the support sample of a given class derive from a similar generating source. An additional insight that points to this direction was given by the later soundification process. Some laughs were recorded by groups of people, while other were generated by an only speaker alone. Moreover, evil laughs that usually starts from the bottom frequencies and tend to reach higher ones were also present. Experiments show that the network is not able to discriminate between an evil laugh and a door opening sounds that similarly slides from lower frequencies to higher ones. *ESC-Net* does not grasp correctly on these frequency-slides, but focuses its attention on the higher spectrum. Listening back to these isolated frequencies for both the correct classifications (Fig. 6.11) and the incorrect ones (Fig. 6.12) no difference was perceived by human ear. Moreover, a human classification would have been impossible by only listening to such audio portions. The soundification process has turned out to be useful for the *church bells* recordings instead: it guided us to understand that the model is paying more attention to the decay between different bell tolls, more than the actual toll sound event itself. Similarly, dry *frog* croaks are much more easy to classify than recordings containing echo or reverb that somehow connects different croaks between them.

6.4.3 Insertion and Deletion

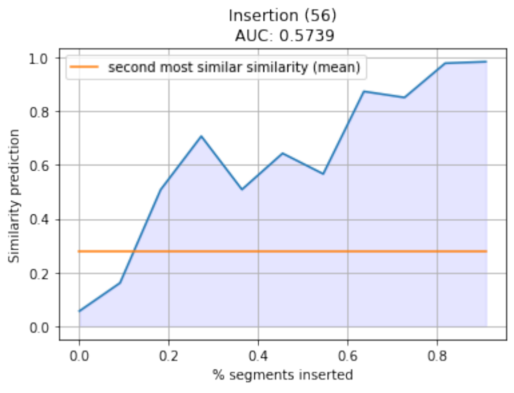
To asses qualitative significance of the explanations found via our perturbation method, we followed a *Randomized input sampling* fashion (Petsiuk, Das, and Saenko 2018) computing the deletion and insertion scores by increasingly deleting and inserting the segments our method value as most important from a full and an empty spectrogram respectively. In this framework, we ideally expect the insertion curve to rapidly increase after only few segments addition, remaining closer to high similarity values as less important segment are gradually added. Additionally, we expect the insertion related *area under curve (AUC)* to be as closer to 1 as possible. On the other hand, we expect the deletion curve to decrease rapidly after only few segments removal, stationing on small similarity values as less important segments are deleted. In this case, we expect the deletion related *area above curve (AAC)* to be as closer to 1 as possible. In both cases, such curves behaviour would imply that our explanation method has indeed found important segments.

To do this, we generated 50 random *5-way one-shot* tasks per each of the 5 test classes and we repeated this process for both dataset. Each of the 50 class-runs would consider that specific class as query label, computing the attribution values for the support set sample belonging to the class in exam using our perturbation-based method. Once the segment attribution values were stored, we then measured:

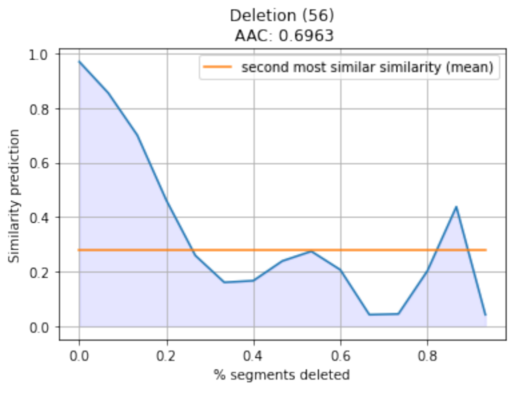
- **Insertion scores** by adding segments from the most to the least important to an empty spectrogram. Similarity prediction between the query sample and such perturbed version of the support set sample was recorded at every segment-addition step.
- **Deletion scores** by removing segments from the most to the least important from the intact support set spectrogram. Similarity prediction between the query sample and such perturbed version of the support set sample was recorded at every segment-removal step.

Finally, we computed the mean value of both *insertion* and *deletion* scores. Additionally, the mean predicted value of the second most similar element of the support sample was also stored for those same 50 one-shot tasks.

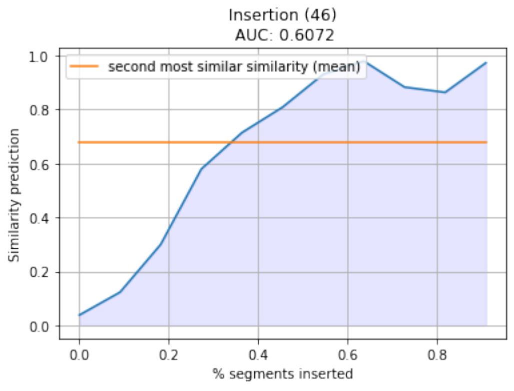
To assess the qualitative relevance of the results we described for the *AudioMNIST* dataset, let us show the insertion and deletion curves for speakers 56 and 46 (Figure 6.13). Both insertion and deletion curves show that the segments our method identifies are somewhat important to the model outcomes. Looking at the speaker 56 insertion curve (Fig. 6.13a), we can see that we only need to add $\sim 0.1\%$ of important segments to let *MNIST-Net* match the query with the correct support



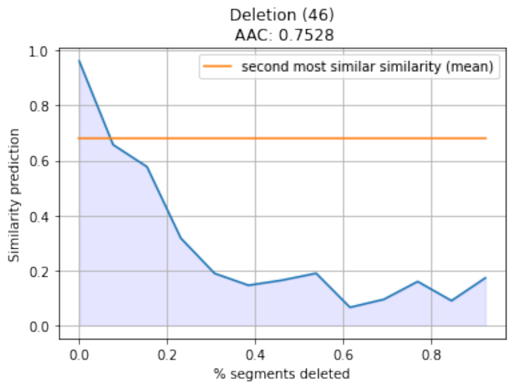
(a) Speaker 56 insertion curve $AUC = 0.5739$



(b) Speaker 56 deletion curve $AAC = 0.6963$



(c) Speaker 46 insertion curve $AUC = 0.6072$

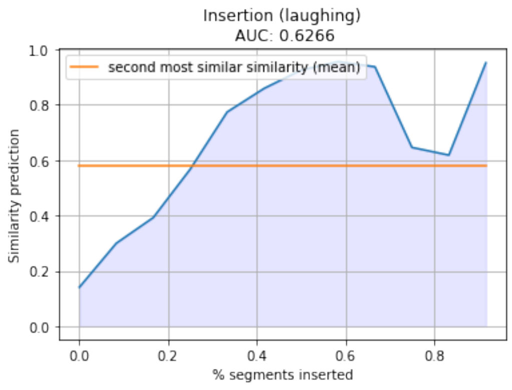


(d) Speaker 56 deletion curve $AAC = 0.7528$

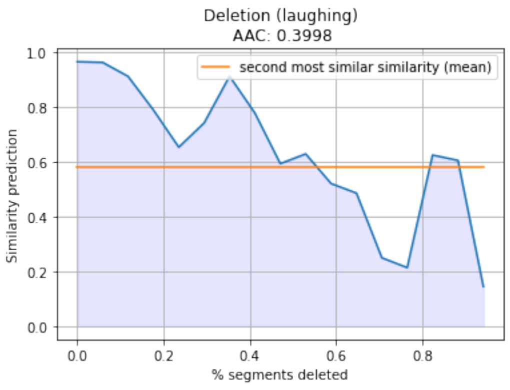
Figure 6.13 *Insertion and deletion curves for speakers 56 and 46*

sample, since the second most similar element scores a mean value of $\sim 0.30\%$. Insertion curve for speakers 46 shows that the average percentage of important segments needed for a correct classification is of $\sim 0.35\%$ instead (Fig. 6.13c). Analyzing the deletion curves we can see that while for speaker 56 we would need to delete in average $\sim 0.25\%$ of important segments to match the right support sample (Fig. 6.13b), speakers 46 would only require a $\sim 0.05\%$ of segments to be removed (Fig. 6.13d). It is due noticing that a strange behaviour is highlighted for the deletion curves of speaker 56. When the percentage of important segments deleted is between $\sim 0.80\%$ to $\sim 0.85\%$, the similarity score increases to values even higher than the second most similar element of the support set.

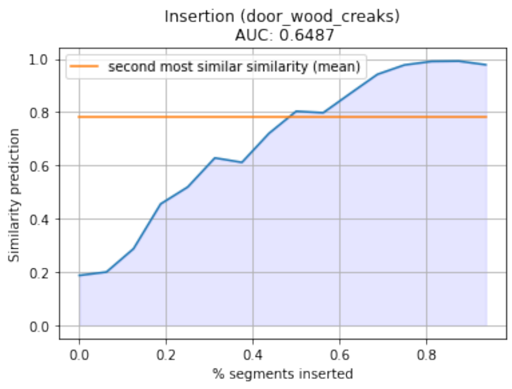
Let us now describe deletion and insertion curves for the *door wood creaks* and *laughing* classes processed by the *ESC-Net* and displayed in Figure 6.14. All curves guide us to think that our explanation methods is somehow capable of pointing to important segments. Both insertion curves imply a reasonably high AUC, despite



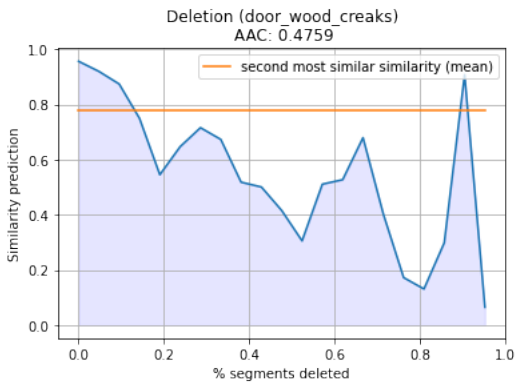
(a) Laughing insertion curve $AUC = 0.6266$



(b) Laughing deletion curve $AAC = 0.3998$



(c) Door wood creaks insertion curve $AUC = 0.6487$



(d) Door wood creaks deletion curve $AAC = 0.4759$

Figure 6.14 *Insertion and deletion curves for laughing and door wood creaks classes*

the fact that the percentage of segment to add to let the network match the query with the right support sample, is of $\sim 0.25\%$ and $\sim 0.5\%$ for the *laughing* (Fig. 6.14a) and *door wood creaks* classes (Fig. 6.14c) respectively. On the other hand, both deletion curves imply AAC lower than 0.5. The percentage of segments to remove for a correct classification is $\sim 0.5\%$ for the *laughing* class (Fig. 6.14b), and $\sim 0.1\%$ for the *door wood creaks* class instead (Fig. 6.14d). The deletion curves of *ESC-Net* let us notice the same weird behaviour we encountered on *MNIST-Net* speaker 56: a peak is always present when more than 80% of important segments are deleted from the starting spectrogram. This finding might lead us to think that both models give importance to silent areas, since they return high similarity values when comparing intact spectrograms with empty ones. This behavior might be due to the fact that the model has never been trained on completely silent recordings, leading to a comparison incapacity when it is presented with such out-of-distribution inputs.

7 Conclusions

In this thesis, we present a novel perturbation-based method to explain Siamese Networks that process audio inputs in the context of *C-way k-shot learning.* Using a perturbation approach based on the spectrogram morphology, the tool evaluates *segments-weighted-average* contribution values to the final outcome considering the interplay of different areas of the overall spectrogram. Such contributions values are computed for every support set sample and then visualized as heatmaps in a shared *min-max* scale. The tool is able to highlight segments important for the model classification process, both towards higher and lower similarity scores. Combining the visual communicative effect and the soundification of such important segments, our approach is able to guide us in understanding both correct and incorrect classification behaviours. In Chapter 6 we reported extensively the experiments conducted on the *AudioMNIST* and the *ESC-50* dataset separately, followed by a *insertion-and-deletion* assessment to validate our proposal.

Experiments on the *AudioMNIST* dataset demonstrate that state-of-the-art performance are achievable in the context of *5-way one-shot* learning by means of Siamese Networks for short audio clips. Moreover, experimental results illustrate that our explanation method can bring some interesting findings to light. Correct classification of female speakers recordings is mainly due to medium-high frequency segments, while their miss-classification depends primarily on segments that reside at the very bottom of the frequency range. A symmetrical behaviour is observed for male speakers audios: a correct classification is usually based on lower frequency values, while incorrect classifications is generally due to segment higher in the frequency spectrum. Our explanation method also brings to surface the model reliance on silent-areas. While in most cases it is present with low magnitudes, there are cases where its importance is higher than actual spoken areas guiding the model towards the wrong prediction.

The experiments carried out on the *ESC-50* dataset confirm that state-of-the-art performance can be reached in the context of *5-way one-shot* learning using Siamese Networks on longer audio clips (Honka 2019). The application of our explanation method extracts different insights, the first one being a strong dependence on the recordings domain range. Such finding might seem trivial, but it has a huge impact on the model generalization capability and an even greater meaning in our *one-shot* learning context. Distinct recordings belonging to the same class label, could still be produced by different sources and in different recording environments. An evil laugh and a people-group laugh are labelled as equals, even though their spectrogram morphology is quite different. The former tend to evolve from low to medium-high frequencies, while the latter mainly reside in the higher frequency spectrum. The network inability of discriminating on medium-high frequency was largely explored analysing correct and incorrect classification for the *laughing* and *door wood creaks* classes jointly. The segments of interest for both classes tend to reside in the higher frequency spectrum, which is where the network has not really learned to discriminate. In these cases, the final decision is deferred to the actual nature of the support set recordings: having crest-shaped values in the mid-high spectrum range is the the only thing that matters. In other cases, the explanations led us to understand the importance of the decay between two sound events that are repeated frequently in the overall recording. Most of the times, decays is the discriminating variable when miss-classification occurs between frog croaks and church bells tools. Despite the good *one-shot* performances, such findings lead to obvious questions towards the model robustness.

A significant conclusion we can draw from using our explanation method is that the recordings domain is crucial to the model trustworthiness. *AudioMNIST* is a pretty simple dataset where every audio clip is only composed of a spoken digit. Moreover, all dataset samples were recorded with the same microphone and in the same environment. If we consider the recordings of a new speaker, they still have the same accent and similar intention when being spoken. On the other hand each *ESC-50* class is composed of a more heterogeneous set of recordings. When queried on these classes, the network might fail to match the two correct samples since they are indeed quite different by nature. Our experiments also showed that the soundification process effectiveness is dataset-dependent: listening back to segments belonging to short audio clips (1 second) is pretty useless since it reconstruct audios that are too short to be semantically meaningful. On the other hand, soundifying portion of longer clips is helpful to the overall explanation goal especially when they include segments residing in distinct frequency ranges.

A critical drawback of our method concerns the time resources needed. Generally, perturbation-based method are computational expensive since they might perturb the input several times. In addition to that, our explanation is linked to a *C-way k-shot* learning framework, and therefore needs to compute contribution values per each of the C elements included in the support set. Another critical aspect of our method is the number of different parameters to tune. In order to have a comprehensive idea of the explanation results each configuration would lead to, one should have to wait for all contribution values to be computed. The hyper-parameter tuning phase could easily become time consuming by itself. A possible mitigation to such problem would imply tuning the parameter settings on a smaller value of C, before running the method on the desired number of classes to explain.

Combination of our tool's generated heatmaps and the insertion-and-deletion assessment, showed model weaknesses on silence-areas. Possible future works could tackle this problem in two separate manners. The data augmentation phase that precedes the spectrogram conversion, could make use of white noise instead of silence-areas. This change could also bring our problem much closer to a real-world scenario where noise is largely present in recording clips. A different solution could make use of completely silenced pairs to train the model. By doing so such areas could be less problematic during our method's perturbations, especially if the networks does not see them as out-of-distribution data points anymore.

Future directions of this work might be refining the explanations, working both on the front-end visualization and the inner algorithm itself. Deeper investigations can be carried out in order to decide which sample to perturb. The experiments we carried out in this thesis keep intact the query sample, to then segment and perturb each support set element based on its own segmentation mask. A different approach could segment and perturb the query sample instead, and measure its distance to each of the support set samples. It would be also interesting to analyze the possibility of perturbing both the query and the support samples together to measure segment-wise similarities. Intriguing aspects worth exploring with additional experiments would be working on different audio conversions, regardless of their ability to get later sonified. Training the Siamese Network on much smaller amount of data could also be explored. This approach would take the problem closer to real-world scenarios where few data repetition are available per each class.

Let us conclude by saying that the motivation behind the explanation method we proposed in this thesis is the desire to explain machine learning models that emulate human learning behaviour. Being able to understand which are the discriminative features for a black-box model that works in a *few-shot* learning framework, could help us understand to which extent machines think as human (at the very least in a context where they are asked to do so). Important aspects to explore with additional experiments would be testing our explanation method on dataset composed of recordings longer than the one we tested in this thesis. Moreover, the combination of heatmaps visualization along with the soundification process, could be further explored on such longer recordings to better understand if it is able to expose segment portions meaningful to the human ear. Despite our proposal was born to explain Siamese Networks, a future research direction would be extending our explanation method on other architectures which are commonly employed in *few-shot* learning tasks like Matching Networks, Relation Networks or Prototypical Networks. Finally, future studies might cover the possibility of exploring our method's efficiency on traditional image inputs.

References

Acconcjaioco, Michelangelo and Stavros Ntalampiras (2021). “One-shot learning for acoustic identification of bird species in non-stationary environments”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 755–762.

Achanta, Radhakrishna et al. (2012). “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11, pp. 2274–2282. DOI: `10.1109/TPAMI.2012.120`.

Adadi, Amina and Mohammed Berrada (2018). “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6, pp. 52138–52160. DOI: `10.1109/ACCESS.2018.2870052`.

Ancona, Marco et al. (2018). “Towards better understanding of gradient-based attribution methods for Deep Neural Networks”. In: *International Conference on Learning Representations*. URL: `https://openreview.net/forum?id=Sy21R9JAW`.

Bach, Sebastian et al. (2015). “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”. In: *PloS one* 10.7, e0130140.

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

Bhati, Saurabhchand et al. (2019). “LSTM Siamese Network for Parkinson’s Disease Detection from Speech”. In: *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1–5. DOI: `10.1109/GlobalSIP45357.2019.8969430`.

Bodria, Francesco et al. (2021). *Benchmarking and Survey of Explanation Methods for Black Box Models*. DOI: `10.48550/ARXIV.2102.13076`. URL: `https://arxiv.org/abs/2102.13076`.

Bromley, Jane et al. (1993). “Signature verification using a” siamese” time delay neural network”. In: *Advances in neural information processing systems* 6.

Chou, Szu-Yu et al. (2019). “Learning to match transient sound events using attentional similarity for few-shot sound recognition”. In: *ICASSP 2019-2019 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 26–30.

Cireşan, Dan, Ueli Meier, and Juergen Schmidhuber (Feb. 2012). “Multi-column Deep Neural Networks for Image Classification”. In: *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* DOI: 10.1109/CVPR.2012.6248110.

Dimitrova, Diana (2020). “The Right to Explanation under the Right of Access to Personal Data: Legal Foundations in and beyond the GDPR”. In: *Eur. Data Prot. L. Rev.* 6, p. 211.

Erhan, Dumitru, Aaron Courville, and Yoshua Bengio (2010). *Understanding representations learned in deep architectures.* Tech. rep. Technical Report 1355, Université de Montréal/DIRO.

Felzenszwalb, Pedro F and Daniel P Huttenlocher (2004). “Efficient graph-based image segmentation”. In: *International journal of computer vision* 59.2, pp. 167–181.

Griffin, D. and Jae Lim (1984). “Signal estimation from modified short-time Fourier transform”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2, pp. 236–243. DOI: 10.1109/TASSP.1984.1164317.

Guidotti, Riccardo et al. (Aug. 2018). “A Survey of Methods for Explaining Black Box Models”. In: *ACM Comput. Surv.* 51.5. ISSN: 0360-0300. DOI: 10.1145/3236009. URL: https://doi.org/10.1145/3236009.

Haenlein, Michael and Andreas Kaplan (2019). “A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence”. In: *California Management Review* 61.4, pp. 5–14. DOI: 10.1177/0008125619864925. eprint: https://doi.org/10.1177/0008125619864925. URL: https://doi.org/10.1177/0008125619864925.

Hershey, Shawn et al. (2016). *CNN Architectures for Large-Scale Audio Classification.* DOI: 10.48550/ARXIV.1609.09430. URL: https://arxiv.org/abs/1609.09430.

Hoffer, Elad and Nir Ailon (2014). *Deep metric learning using Triplet network.* DOI: 10.48550/ARXIV.1412.6622. URL: https://arxiv.org/abs/1412.6622.

Honka, Tapio (2019). “One-shot Learning with Siamese Networks for Environmental Audio”. In.

Hooker, Sara et al. (2018). *A Benchmark for Interpretability Methods in Deep Neural Networks.* DOI: 10.48550/ARXIV.1806.10758. URL: https://arxiv.org/abs/1806.10758.

Hu, Linwei et al. (2018). *Locally Interpretable Models and Effects based on Supervised Partitioning (LIME-SUP).* DOI: 10.48550/ARXIV.1806.00663. URL: https://arxiv.org/abs/1806.00663.

Irving, Benjamin (2016). *maskSLIC: Regional Superpixel Generation with Application to Local Pathology Characterisation in Medical Images.* DOI: 10.48550/ARXIV.1606.09518. URL: https://arxiv.org/abs/1606.09518.

Ivanovs, Maksims, Roberts Kadikis, and Kaspars Ozols (2021). “Perturbation-based methods for explaining deep neural networks: A survey”. In: *Pattern Recognition Letters* 150, pp. 228–234. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2021.06.030. URL: https://www.sciencedirect.com/science/article/pii/S0167865521002440.

Khosla, Prannay et al. (2020). *Supervised Contrastive Learning.* DOI: 10.48550/ARXIV.2004.11362. URL: https://arxiv.org/abs/2004.11362.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization.* DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

Koch, Gregory, Richard Zemel, Ruslan Salakhutdinov, et al. (2015). “Siamese neural networks for one-shot image recognition”. In: *ICML deep learning workshop.* Vol. 2. Lille.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems.* Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Lecun, Y. et al. (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Lundberg, Scott and Su-In Lee (2017). *A Unified Approach to Interpreting Model Predictions*. DOI: 10.48550/ARXIV.1705.07874. URL: https://arxiv.org/abs/1705.07874.

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

Miller, Tim (2017). *Explanation in Artificial Intelligence: Insights from the Social Sciences*. DOI: 10.48550/ARXIV.1706.07269. URL: https://arxiv.org/abs/1706.07269.

Mishra, Saumitra, Bob L. Sturm, and Simon Dixon (2017). "Local Interpretable Model-Agnostic Explanations for Music Content Analysis". In: *ISMIR*.

Petsiuk, Vitali, Abir Das, and Kate Saenko (2018). *RISE: Randomized Input Sampling for Explanation of Black-box Models*. DOI: 10.48550/ARXIV.1806.07421. URL: https://arxiv.org/abs/1806.07421.

Piczak, Karol J. (2015). "Environmental sound classification with convolutional neural networks". In: *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. DOI: 10.1109/MLSP.2015.7324337.

Ras, Gabrielle, Erdi Çallı, and Marcel van Gerven (2022). *Hermitry Ratio: Evaluating the validity of perturbation methods for explainable deep learning*. URL: https://openreview.net/forum?id=vQ58AMOw4Il.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. DOI: 10.48550/ARXIV.1602.04938. URL: https://arxiv.org/abs/1602.04938.

Selvaraju, Ramprasaath R. et al. (Oct. 2019). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2, pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

Shankaranarayana, Sharath M. and Davor Runje (2019). *ALIME: Autoencoder Based Approach for Local Interpretability*. DOI: 10.48550/ARXIV.1909.02437. URL: https://arxiv.org/abs/1909.02437.

Shih, Chin-Hong et al. (2017). "Investigating Siamese LSTM networks for text categorization". In: *2017 Asia-Pacific Signal and Information Processing As-*

sociation Annual Summit and Conference (APSIPA ASC), pp. 641–646. DOI: 10.1109/APSIPA.2017.8282104.

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje (2017). "Learning Important Features Through Propagating Activation Differences". In: DOI: 10.48550/ARXIV.1704.02685. URL: https://arxiv.org/abs/1704.02685.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman (2013). *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.* DOI: 10.48550/ARXIV.1312.6034. URL: https://arxiv.org/abs/1312.6034.

Simonyan, Karen and Andrew Zisserman (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition.* DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556.

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). "Prototypical networks for few-shot learning". In: *Advances in neural information processing systems* 30.

Sung, Flood et al. (June 2018). "Learning to Compare: Relation Network for Few-Shot Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Tang, Wensi, Lu Liu, and Guodong Long (2020). *Interpretable Time-series Classification on Few-shot Samples.* DOI: 10.48550/ARXIV.2006.02031. URL: https://arxiv.org/abs/2006.02031.

Traunmüller, Hartmut and Anders Eriksson (1995). "The frequency range of the voice fundamental in the speech of male and female adults". In: *Unpublished manuscript* 11.

Utkin, Lev, Maxim Kovalev, and Ernest Kasimov (Jan. 2020). "Explanation of Siamese Neural Networks for Weakly Supervised Learning". In: *Computing and Informatics* 39, pp. 1172–1202. DOI: 10.31577/cai_2020_6_1172.

Vélez, Ivette, Caleb Rascon, and Gibrán Fuentes-Pineda (2018). "One-shot speaker identification for a service robot using a cnn-based generic verifier". In: *arXiv preprint arXiv:1809.04115.*

Vinyals, Oriol et al. (2016). "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems.* Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

Wang, Yaqing et al. (June 2020). “Generalizing from a Few Examples: A Survey on Few-shot Learning”. In: *ACM Computing Surveys* 53, pp. 1–34. DOI: 10.1145/3386252.

Wang, Yu et al. (2020). “Few-Shot Sound Event Detection”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 81–85. DOI: 10.1109/ICASSP40776.2020.9054708.

Wexler, James et al. (2020). “The What-If Tool: Interactive Probing of Machine Learning Models”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1, pp. 56–65. DOI: 10.1109/TVCG.2019.2934619.

Wolters, Piper et al. (2020). *A Study of Few-Shot Audio Classification*. DOI: 10.48550/ARXIV.2012.01573. URL: https://arxiv.org/abs/2012.01573.

Zafar, Muhammad Rehman and Naimul Mefraz Khan (2019). *DLIME: A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems*. DOI: 10.48550/ARXIV.1906.10263. URL: https://arxiv.org/abs/1906.10263.

Zeiler, Matthew D and Rob Fergus (2013). *Visualizing and Understanding Convolutional Networks*. DOI: 10.48550/ARXIV.1311.2901. URL: https://arxiv.org/abs/1311.2901.

Zhang, Yichi, Bryan Pardo, and Zhiyao Duan (2019). “Siamese Style Convolutional Neural Networks for Sound Search by Vocal Imitation”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.2, pp. 429–441. DOI: 10.1109/TASLP.2018.2868428.

Acknowledgements

Many thanks to *Riccardo Guidotti* for sharing his inspirational ideas and expertise. Thank you for your constant and essential mentorship throughout the entire course of this thesis.