UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

MARCEL POZZOBON BORIN

CLASSIFICAÇÃO DE RUÍDO AMBIENTAL: DA CONSTRUÇÃO DE UMA BASE DE DADOS À MODELAGEM EM APRENDIZAGEM PROFUNDA

ENVIRONMENTAL NOISE CLASSIFICATION: FROM DATASET CONSTRUCTION TO DEEP LEARNING MODELLING

Campinas

2024

MARCEL POZZOBON BORIN

ENVIRONMENTAL NOISE CLASSIFICATION: FROM DATASET CONSTRUCTION TO DEEP LEARNING MODELLING

CLASSIFICAÇÃO DE RUÍDO AMBIENTAL: DA CONSTRUÇÃO DE UMA BASE DE DADOS À MODELAGEM EM APRENDIZAGEM PROFUNDA

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Telecomunicações e Telemática

Dissertation presented to the Faculty of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Telecommunications and Telematics

Supervisor: Prof. Dr. Bruno Sanches Masiero

Co-orientador: Prof. Dra. Carolina Rodrigues Alves Monteiro

ESTE TRABALHO CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO MARCEL POZZOBON BORIN, E ORIENTADA PELO PROF. DR. BRUNO SANCHES MASIERO

Campinas

2024

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

Borin, Marcel Pozzobon, 1991-
B644e Environmental noise classification : from dataset construction to deep learning modelling / Marcel Pozzobon Borin. – Campinas, SP : [s.n.], 2024.

Orientador(es): Bruno Sanches Masiero.
Coorientador(es): Carolina Rodrigues Alves Monteiro.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Faculdade de Engenharia Elétrica e de Computação.

1. Processamento digital de sinais. 2. Aprendizado de máquina. 3. Redes neurais convolucionais. 4. Acústica. 5. Ruído urbano. I. Masiero, Bruno Sanches, 1981-. II. Monteiro, Carolina Rodrigues Alves. III. Universidade Estadual de Campinas (UNICAMP). Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações complementares

Título em outro idioma: Classificação de ruído ambiental : da construção de uma base de dados à modelagem em aprendizagem profunda
Palavras-chave em inglês:
Digital signal processing
Machine learning
Convolutional neural networks
Acoustics
Urban noise
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Bruno Sanches Masiero [Orientador]
Rafael Ferrari
Júlio Apolinário Cordioli
Data de defesa: 15-10-2024
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-0601-1635
- Currículo Lattes do autor: http://lattes.cnpq.br/6600673642253347

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Marcel Pozzobon Borin **RA:** 263789

Data da Defesa: 15 de outubro de 2024

Título da Dissertação: Classificação de ruído ambiental: da construção de uma base de dados à modelagem em aprendizagem profunda.

Prof. Dr. Bruno Sanches Masiero (Presidente, Universidade Estadual de Campinas)
Prof. Dr. Rafael Ferrari (Universidade Estadual de Campinas)
Prof. Dr. Júlio Apolinário Cordioli (Universidade Federal de Santa Catarina)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

Resumo

Com o crescimento de grandes cidades e aglomerações em centros urbanos, o ruído tem se tornado cada vez mais crítico nessas áreas. Entre as fontes sonoras geradoras de incômodo para a população, encontram-se especialmente o tráfego de veículos e o ruído de obra. Em São Paulo, os ruídos emitidos devem respeitar leis e normas vigentes, como a ABNT NBR 10151, e o decreto municipal Nº 60.581 que traz limites para ruídos de obras. Como forma de gerenciamento e mitigação de ruído urbano, o monitoramento sonoro contínuo tornou-se uma ferramenta importante, especialmente quando aplicado em conjunto com modelos de identificação automática de fontes sonoras, que reduz drasticamente esforços manuais humanos e ajuda na tomada de decisões de gerenciamento de ruído. Contudo, uma correta classificação de sons urbanos pode ser uma tarefa árdua pela complexidade das paisagens sonoras urbanas, que pode conter muitas fontes sonoras simultâneas e ruído atrelado. Além disso, é necessária uma base de dados de sons ampla e representativa para o ambiente a ser monitorado. Este trabalho, primeiramente, se propõe a construção de uma base de dados de sons urbanos para a cidade de São Paulo, Brasil, com as principais fontes sonoras provenientes dessa área. Em seguida, são propostas arquiteturas de redes neurais capazes de classificar corretamente esses sons, entre elas estão modelos CNN, RNN e híbridos. Este último atingiu os melhores resultados nas métricas avaliadas. Sons de baixas frequências, especialmente veículos, apresentaram uma maior dificuldade em serem corretamente identificados. Neste estudo, foi utilizado o Log-Mel-spectrogram como dado de entrada em 8 combinações diferentes de resoluções no tempo e frequência. Ainda foram utilizadas técnicas de aumento de dados e modelo de classificação hierárquica em conjunto com o modelo híbrido, estas, porém, não conseguiram demonstrar melhoria significativa nas métricas de classificação. Trabalhos futuros devem focar especialmente, na otimização dos resultados de algumas fontes sonoras de baixas frequências, através de novos modelos e pré-processamentos.

Palavras-chaves: base de dados de sons; classificação de sons urbanos; redes neurais convolucionais; redes neurais híbridas; aumento de dados; classificação hierárquica

Abstract

Noise has become increasingly critical in urban areas, due to the growth of large cities and urban agglomerations. Among the noise sources causing inconvenience to the population are especially vehicle traffic and construction noise. In São Paulo, emitted noises must comply with current laws and regulations, such as ABNT NBR 10151, and the municipal decree No. 60.581, which sets limits for construction noise. As a method of managing and mitigating urban noise, continuous sound monitoring has become an important tool, especially when employed together with models for automatic identification of sound sources, which drastically reduces human manual efforts and aids in noise management decision-making. However, the correct classification of urban sounds can be an arduous task due to the complexity of urban soundscapes, which can contain many simultaneous sound sources and associated noise. Additionally, a broad and representative sound dataset is required for the environment to be monitored. This work first proposes the construction of an urban sound database for the city of São Paulo, Brazil, with the main sound sources from this area. Subsequently, neural network architectures capable of correctly classifying these sounds are proposed, including CNN, RNN, and hybrid models. The latter achieved the best results in the evaluated metrics. Low-frequency sounds, especially vehicles, showed greater difficulty in being correctly identified. In this study, the Log-Mel-spectrogram was used as input data in 8 different combinations of time and frequency resolutions. Data augmentation techniques and hierarchical classification models were also used together with the hybrid model, however, they did not demonstrate significant improvement in classification metrics. Future research should specially address improving certain low-frequency sound sources classification, focusing on new models and preprocessing techniques.

Keywords: sound dataset; urban sound classification; convolutional neural networks; hybrid neural networks; data augmentation; hierarchical classification

"A future is not given to you. It is something you must take for yourself."
(NieR: Automata)

List of Figures

List of Tables

Contents

1 Introduction

Urban areas are home to more than half of the global population, and it is predicted that two-thirds of the world's population will live in urban agglomerations in the next 30 years (RITCHIE; ROSER, 2018). According to the European Environment Agency (EEA) (2020), air pollution is the most harmful environmental exposure to public health, followed by noise pollution. For this reason, metropolitan areas have become the epicenter of this hazard. Construction sites are one of the main sources of noise, along with traffic noise. In São Paulo, Brazil, whose population is estimated to be over 12 million residents, noise complaints have increased by 41% from 2022 to 2023 (G1 SP; TV GLOBO, 2023). However, the Brazilian mega-city is still in the early stages of dealing with noise pollution.

In 2016, the city of São Paulo approved Law 16.499, which determines the creation of a noise map by 2030. This map will work as a tool to assist public agencies and society in establishing measures to improve the quality of life of São Paulo residents regarding noise pollution. In addition, a recent regulatory action has been decreed to mitigate noise pollution in response to disturbance from construction noise. This action, outlined in Decree No. 60.581 (SÃO PAULO, 2021b), establishes sound pressure level limits for both daytime and nighttime generated by construction sites in the city. São Paulo has undergone significant verticalization since the approval of the new city's Master Plan in 2013, and from 2021 to 2022, 721 new construction licenses have been approved (MOTA; CAMILLA, 2021), and 1363 demolition permits have been issued (SÃO PAULO, 2021a).

In 2002, the European Parliament and Council adopted Directive 2002/49 on the assessment and management of environmental noise. The management approach is based on diagnosis and specific actions. A common strategy in urban noise mitigation is the creation of noise maps, which are essential diagnostic tools that help estimate how many people are exposed to high noise levels. However, they fail to provide a complete picture of the urban soundscape (KLOTH *et al.*, 2008).

Nevertheless, noise maps are often criticized for their drawbacks, including being time-consuming, expensive, and static. Therefore, real-time sound monitoring emerges as a viable solution, offering advantages such as continuous data collection over long periods and dynamic noise maps using online platforms. In addition to measuring noise levels, understanding the sound source is fundamental for understanding the soundscape and the possible causes of noise pollution.

The relevant standards and legislation must be applied to verify compliance with noise limits. Long-term noise monitoring is often the best practical approach to investigate sound levels. However, urban soundscapes can be highly complex due to the multiple possible sound sources that may happen simultaneously. Therefore, automatic identification algorithms of sound events can be hugely time-saving and an excellent tool for better understanding the soundscape.

In recent years, there has been an increasing interest in studies focused on environmental sound classification (ESC), especially in identifying specific sound events under interest. These studies have been applied in various practical domains, such as robotic hearing, smart home systems, audio monitoring, and soundscape assessment. Unlike regular and structured sounds such as speech and music, environmental sounds usually lack specific time patterns, like melodies or rhythms, or semantic sequences, like phonemes. Consequently, identifying universal features that can accurately represent the diverse temporal patterns of environmental sounds is often challenging. Moreover, environmental sounds usually contain noise and other irrelevant simultaneous sounds, resulting in complex structures characterized by high variability, diversity, and unstructured characteristics (ZHANG *et al.*, 2021).

The growing interest in the sound analysis of urban environments was partially facilitated by sensor networks and the abundance of online multimedia content containing urban scenes. However, while research in related fields such as speech, music, and bioacoustics is extensive, there is a relative scarcity of work focused on analyzing urban acoustic environments. Moreover, existing research in this area primarily concentrates on classifying auditory scene types, such as streets or parks, rather than identifying specific sound sources within these scenes, such as vehicles, machines, and animals (SALAMON *et al.*, 2014).

To tackle these tasks, researchers have explored multiple methods in signal processing and machine learning to work with sound classification and detection problems. Traditional approaches often involve separating the analysis into different feature representation and classification steps, requiring manual operations such as extracting the Mel-frequency cepstral coefficient (MFCC), Mel-spectrum features, or wavelet transforms. These methods have the disadvantages of the necessity for extensive manual work and multiple experiments to find the best features (MU *et al.*, 2021).

In contrast, deep neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have emerged as promising solutions for sound classification. CNNs excel at capturing time-frequency features, while RNNs effectively learn time-dependent patterns in audio features. So, hybrid models such as CNN-RNN (CRNN) have shown good performance in tasks requiring both spatial and temporal fea-

ture learning, as needed in image detection and speech recognition (ACHARYA; BASU, 2020).

The large effort involved in manually annotating real-world data means that datasets based on field recordings tend to be relatively small. However, a solid and representative dataset is crucial to achieve accurate performance from a supervised model (ROSEBROCK, 2017b). As a representative dataset, the training data should be as similar as possible to the real-world application, which generally includes data with high background noise or sound sources highly modified by the propagation path in the case of environmental noise. The currently available free environmental audio datasets present multiple important sound sources. However, they still lack some important sound sources or are not entirely representative of the Brazilian soundscape in São Paulo. Many existing urban noise monitoring datasets fail to accurately depict this issue's complexities. These datasets typically feature recordings sourced from Freesound (FONT *et al.*, 2013), lacking the authenticity of real-world noise monitoring scenarios. Furthermore, while some datasets include audio recordings captured in real urban environments, they often offer limited label sets, typically focused on human sounds and traffic. A description of the current main datasets addressing urban sounds is provided in Section 3.1.

Tools like deep learning models and extensive audio datasets are integral to a rapidly growing area of machine perception known as Machine Listening, which can be considered the auditory counterpart to computer vision. In this field, a combination of signal processing techniques and machine learning systems is used to extract meaningful information from sounds and perform tasks, such as the audio classification conducted in this project (LYON, 2017).

1.1 Objective

The primary goal of this study is to develop a comprehensive sound dataset representative of São Paulo's urban environment, designed to identify a broad range of sounds that may exceed noise regulations. This includes common intrusive sounds such as traffic noise, natural sounds, as well as human and machine-generated noises. Following the dataset creation, the study will focus on building several deep learning models, including convolutional neural networks, recurrent networks, and hybrid approaches, to accurately classify the sounds in the dataset. These models will be evaluated and compared using relevant performance metrics.

1.2 Work organization

The thesis is divided into five chapters, each addressing a part of the research. Chapter 1 introduces the topic of urban sound classification and highlights the importance of applying machine learning to urban noise management. It also addresses the main challenges in sound classification and provides an overview of the current state of development in the area. Chapter 2 is the theoretical background, which provides all the essential information and concepts for machine learning and deep learning, focusing on audio classification. This chapter also discusses two important architectures: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Additionally, it includes an explanation of the signal-processing techniques used in this work. Chapter 3 outlines the methodology. Firstly, this chapter introduces the main urban sound datasets that are freely available. Then, it details how the construction of the dataset from this work was carried out and the machine learning architecture implemented. It also describes the configurations of the dataset preprocessing considered in the experiments. Chapter 4 presents the results and their discussion, analyzing the performance of various configurations and models explored in the thesis. Finally, the last chapter concludes the thesis with the main points and conclusions explored during the work and suggestions for future works.

2 Theoretical Background

This chapter provides an overview about the fundamentals of machine learning, signal processing and the neural network architectures used in this project under the context of urban sound classification.

2.1 Machine Learning Principles

Machine learning (ML) can be defined as the process of addressing a problem through two main steps: collecting a dataset from a real-world application and constructing a statistical model algorithm trained on that dataset, usually applied in a way to solve a practical problem (VEMURI, 2020). Machine learning techniques have facilitated significant progress in automating data processing and improving pattern recognition capabilities across a wide range of fields, including computer vision, image processing, speech analysis, and physical sciences. In the realm of acoustics, machine learning is a quickly evolving field, offering a large number of promising solutions to address the acoustic challenges mentioned earlier (BIANCO *et al.*, 2019).

Machine Learning systems can be categorized based on the level and type of supervision they receive during the training. There are four main categories of machine learning models: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement Learning.

In supervised learning, the training dataset used as input to the algorithm includes predefined solutions, referred to as labels, with classification being a typical application of this method. In this context, the dataset comprises a collection of labeled samples denoted as $\{(x_i, y_i)\}_{i=1}^{N}$, where each element x_i, from N instances, represents a feature vector, and y_i is the respective label. This feature vector is a multidimensional vector, with each dimension indexed as $j = 1, \ldots, D$, containing a value that characterizes the respective example. These values are called features and are denoted as $x^{(j)}$.

In unsupervised learning, the dataset consists of unlabeled examples $\{x_i\}_{i=1}^{N}$, where each x represents a feature vector. The objective of unsupervised learning is to develop a model that takes x as input and transforms it into another vector or a value useful for problem-solving. For instance, in clustering, the model assigns each feature vector to a cluster. In dimensionality reduction problems, it produces a feature vector with fewer dimensions than the input, while in outlier detection tasks, it outputs a real number indicating the deviation of x from a typical example in the dataset.

In semi-supervised learning, the dataset includes both labeled and unlabeled examples, typically with a much larger quantity of unlabeled data. The objective of a semi-supervised learning algorithm is similar to the supervised learning algorithm, with the expectation that the large amount of unlabeled data can aid in the creation of an improved model.

Reinforcement learning is a type of machine learning where a machine interacts with an environment, makes decisions and learns by receiving rewards for its actions. The goal is to find the best way to act in different situations to maximize rewards. The objective of a reinforcement learning algorithm is to learn a policy, represented by a function f, which, similar to a supervised learning model, takes a state's feature vector as input and outputs the optimal action that maximizes expected average rewards (GERON, 2019).

2.2 Classification and Regression

Supervised machine learning is divided into two primary types: classification and regression models (PAJANKAR; JOSHI, 2022).

2.2.1 Classification

This first type of model automatically assigns a label to an unlabeled example, for instance, predicting if the image belongs to labels "cat", "dog" or "horse". Classification algorithms use labeled examples to train and build models that can assign labels to new, unseen, unlabeled data. A classification model can also be used, for example, to classify sounds that are recorded. The classification task can also be divided into:

- **Binary Classification:** It refers to those classification tasks that only have two classes. For example, classifying between a cat or a dog.
- **Multi-class:** It refers to those classification tasks that have more than two classes. For example, classifying a sample as a cat, dog", bird, or bear.
- **Multi-label Classification:** It refers to those classification tasks that have two or more classes for a sample, where one or more classes may be predicted. For example, a picture containing both a cat and a dog.

2.2.2 Regression

In regression, the goal is to predict a value label from a continuous range, such as estimating house prices based on features like size, number of bedrooms, and location. Thus, regression models provide numerical predictions for unlabeled data.

2.3 Hierarchical Classification

This section is an overview of the main hierarchical classification approaches from Silla and Freitas (2011). Hierarchy is commonly used when there is a taxonomy of classes that can be grouped by similar characteristics.

2.3.1 Flat Classification

The flat classification approach is the most common method for dealing with hierarchical classification problems. It works by ignoring the class hierarchy and predicting only the leaf node classes. So, this approach is similar to traditional classification algorithms during the training and testing stages. However, it still indirectly addresses the hierarchical structure of the problem by implicitly assigning all the ancestors of a leaf node to the same class as well. The main drawback of this approach is that it requires building a classifier to distinguish among a possibly large number of leaf classes; therefore, it does not take advantage of the parent-child class relationships from the hierarchy. Figure 2.1 shows the representation of a flat classification approach.

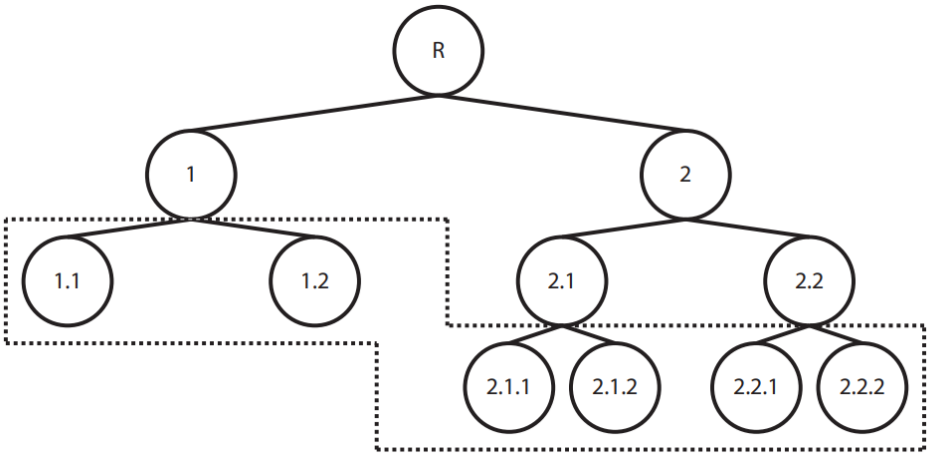


Figure 2.1 – Flat classifier representation from Silla and Freitas (2011).

2.3.2 Local classifiers

The local classifiers can also be divided into three approaches: Local Classifier Per Node, Local Classifier Per Level, and Local classifier per parent Node.

2.3.2.1 Local Classifier Per Node

The local classifier per node approach involves training individual binary classifiers for each node in the class hierarchy. This approach uses multi-class or binary classifiers like the One-Against-One scheme for Binary SVMs to distinguish between child

nodes for each parent node in the hierarchy. Figure 2.2 shows the representation of a local classifier per node approach.

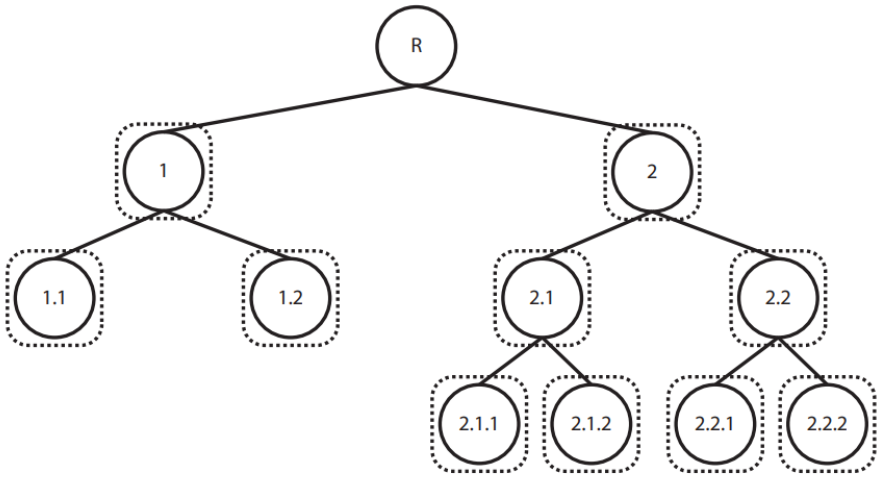


Figure 2.2 – Local classifier per node representation from Silla and Freitas (2011).

2.3.2.2 Local Classifier Per Level

The local classifier per level approach works by training a single multi-class classifier for each level in the class hierarchy. Figure 2.3 shows the representation of a local classifier per level approach.

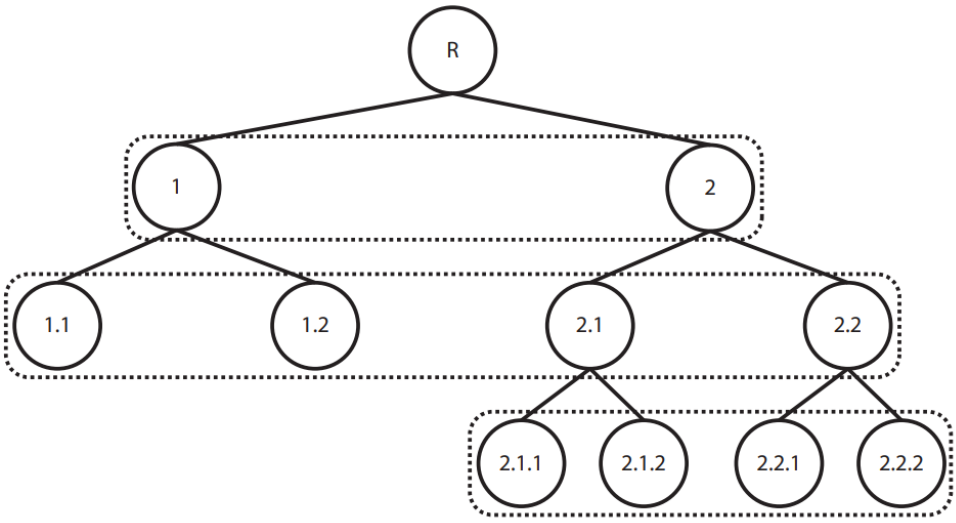


Figure 2.3 – Local classifier per level representation from Silla and Freitas (2011).

2.3.2.3 Local classifier per parent Node

The local classifier per parent node approach involves training a multi-class classifier for each parent node to distinguish among its child nodes. Figure 2.4 shows the representation of a local classifier per parent node approach.

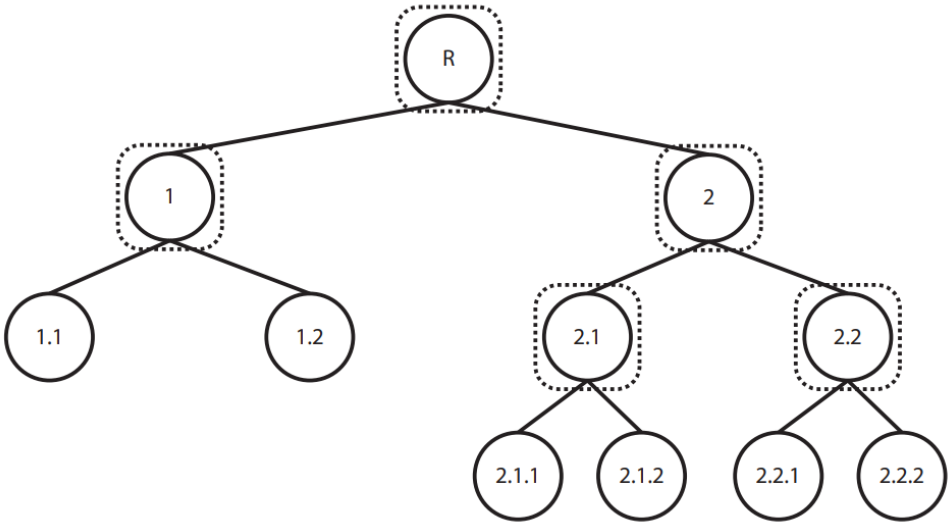


Figure 2.4 – Local classifier per parent node representation from Silla and Freitas (2011).

2.3.3 Big-bang (or Global Classifier)

While local approaches are used to address hierarchical classification, a global classifier offers the advantage of significantly reducing the total size of the classification model compared to the local models used in the local classifier approaches. In the global classifier approach, a single and more complex model is created from the training data, considering the entire class hierarchy in a single algorithm run. During the testing phase, this model can assign classes at multiple levels of the hierarchy to test examples. Figure 2.5 shows the representation of the global classifier approach.

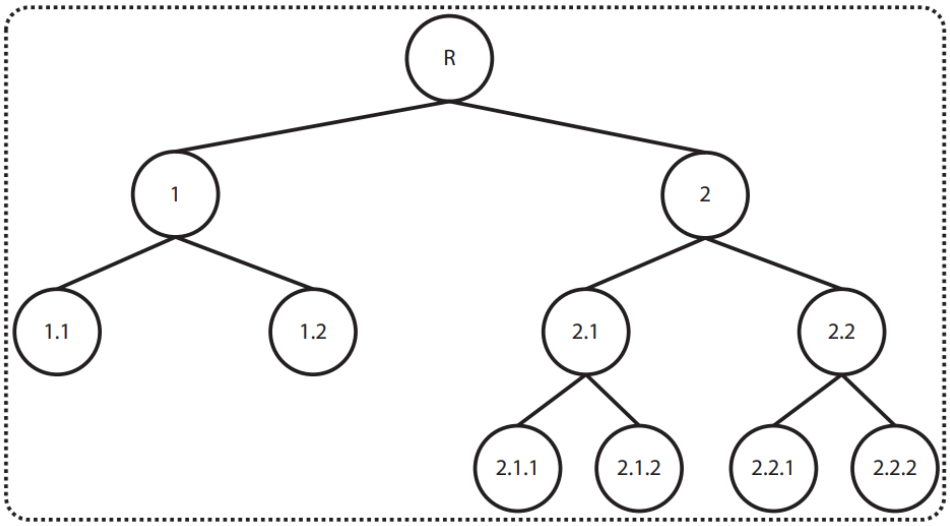


Figure 2.5 – Global classifier representation from Silla and Freitas (2011).

2.4 Machine Learning and Data Fitting

It is crucial to understand how to approach a machine learning problem, as the complexity of the data plays a key role in selecting the most appropriate model. Models perform optimally when specifically designed to address the given task effectively. Two situations can arise when there is a mismatch between model and data complexity. When a high-capacity model is used for a low-complexity task and dataset, an overfitting situation will occur, in which the model learns too many details from specific training data and cannot generalize to other unseen samples. The opposite is also possible; when the task is too complex for a low-capacity model, an underfitting behavior will be observed since the model cannot learn the correct relationships within the data. Both situations must be avoided, so a well-suited model should be chosen based on the dataset complexity (BIANCO *et al.*, 2019). Figure 2.6 shows a graphical representation of the underfitting and overfitting of a model in a dataset.

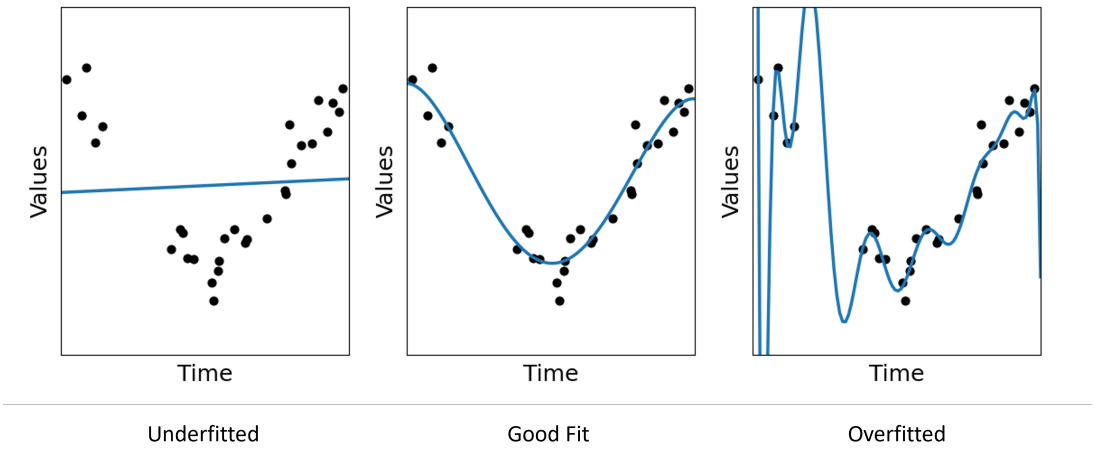


Figure 2.6 – Example of underfitting, good fitting, and overfitting situation. Source: Author.

2.5 Parameterized Learning

Parameterization is the process of defining the required parameters for a specific model. In the context of machine learning, this process involves defining a problem based on four fundamental elements: data, a scoring function, a loss function, and parameters such as weights and biases.

Data (x_i) is the fundamental component that constitutes the learning input. It comprises the data points, such as raw pixel intensities from images or extracted features and their corresponding class labels. The scoring function (f_m) is paramount, as it maps the input data to class labels using mathematical functions. For instance, in the case

of images, this function processes the data points (pixels) as input, and it outputs the predicted class labels. The loss function is a metric for evaluating our predicted class labels' similarity to the actual ground-truth labels. The goal is to achieve a high similarity between these label sets, resulting in a lower loss value, which means a higher classification accuracy, especially within the training set. Finally, the classifier's Weight Matrix and Bias Vector (typically represented as W and b) are the elements that are actively optimized during training. Both outputs from the scoring and loss functions are used to adjust these values to enhance classification accuracy.

While additional parameters may depend on the model, these four elements are the most common aspects of parameterized learning. For this reason, defining these components enables us to apply optimization techniques to determine an optimal set of parameters (W and b) that minimize the loss function concerning the scoring function, thereby boosting classification accuracy on our dataset (ROSEBROCK, 2017b).

A simple linear mapping would be expressed as:

$$f_m(x_i, W, b) = Wx_i + b. \tag{2.1}$$

2.6 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a type of binary classifier that seeks to find an optimal hyperplane (Figure 2.7) that separates different classes of data points with the maximum margin (distance between the hyperplane and the observations closest to the hyperplane). This method is based on the concept of structural risk minimization (SRM), which aims to minimize the generalization error on unseen data instead of minimizing the training error like traditional methods.

With an optimal hyperplane defined by W and b, the decision function $f(x)$ for classifying a new, unknown data point x is given by:

$$f(x) = \text{sign}\,(Wx + b) = \text{sign}\left(\sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{x}_i, x) + b\right), \tag{2.2}$$

where N_S is the number of support vectors, $\mathbf{x}_i$ are the support vectors, α_i are the Lagrange multipliers, y_i indicates the label (± 1) of the support vectors, $K(\mathbf{x}_i, x)$ is the kernel function that computes the similarity between the support vectors and the new data point x, "sign" is the sign function that determines the class label based on the computed value.

The goal of the SVM method is to create a decision boundary that correctly

classifies as many new unseen data points as possible while maximizing the margin between different classes. This makes SVM a popular choice for studies in which tasks like classification and regression are conducted due to its capability to handle complex datasets and make robust predictions (WANG *et al.*, 2006). There are two types of SVM algorithms:

- **Linear SVM** is a model in which a hyperplane separates data points into two classes using a straight line. For multiclass problems, it usually uses the "One-vs-Rest" strategy, which consists of fitting one classifier per class. For each classifier, the class is fitted against all the other classes.
- **Non-Linear SVM** is a model used when the data cannot be effectively separated by a linear hyperplane in the original feature space. In these cases, class overlap can occur, and advanced techniques like kernel tricks should be employed to classify the data. In real-world applications, data is not usually easily separable by straight lines, necessitating kernel tricks to effectively solve them. Using a kernel function, SVMs can implicitly map input data into a higher-dimensional space where a linear separator (hyperplane) can be used to divide the classes. Some examples of non-linear kernels are polynomial, sigmoid, radial basis function (RBF), Bessel, Anova, etc.

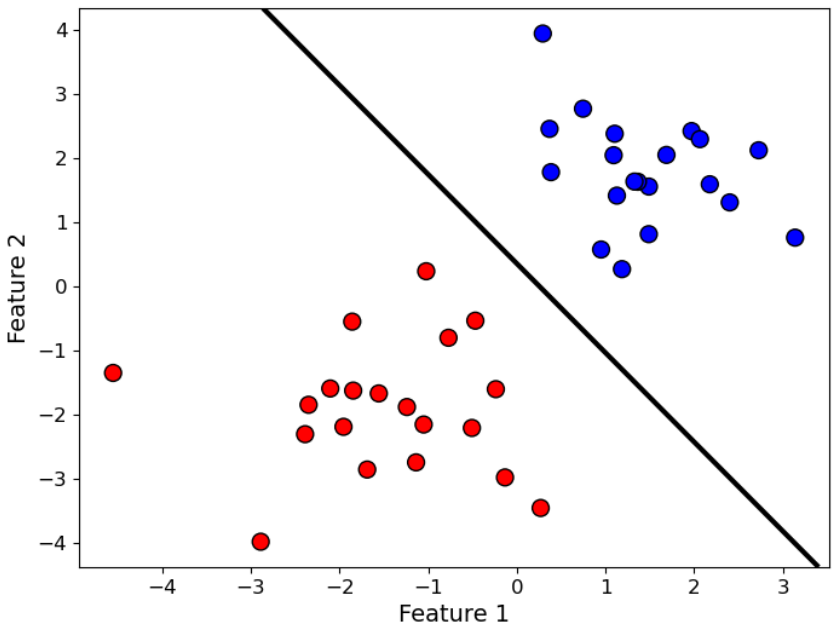


Figure 2.7 – Example of a hyperplane generated by linear SVM classification. Source: Author.

2.7 Artificial Neural Networks (ANN)

Artificial Neural Networks are a machine learning method inspired by the human brain composed of stacks of inter-connected neuron blocks or layers. Each neuron has a weight that is iteratively updated by the input data during the training phase, usually through an algorithm designed to minimize the difference between the predicted and actual output. Feed-forward neural networks (FNN) consist of sequential layers of fully connected (FC) neurons, and each neuron's output is passed as input to every neuron in the next layer without any feedback. This can be schematically seen in Figure 2.8. The example shows an input layer, an output layer, and two intermediate layers, also known as hidden layers. The ANNs with more than one hidden layer are often called Deep Neural Network (DNN) and are frequently used in state-of-the-art models for audio classification (PUENTE, 2018).

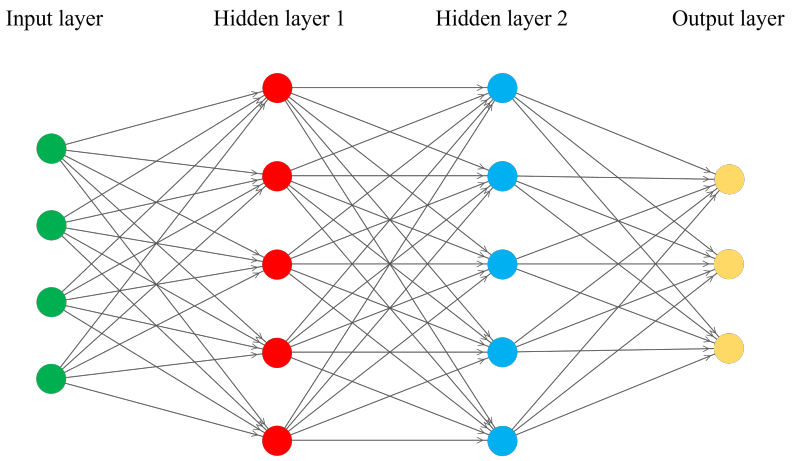


Figure 2.8 – Deep learning architecture using two hidden layers. Source: Author.

Figure 2.9 shows the mathematical relation between the inputs and output of a single neuron. The first layer is a block of n neurons connected to a single neuron in the next layer. Each neuron has an x_n value that is multiplied by its respective weight (w_n). The products are summed into a single number, and an additional bias (b) is incorporated. In summary, the output is a weighted sum of the input, thus a linear combination. A possibly non-linear activation function (f) is applied to the result to learn a more complex relationship between input and output, improving its capacity for learning non-linear behavior (CAKIR, 2019). A rectified linear unit (ReLU) function, for example, is the most popular non-linear activation in recent years, and it is defined as $f(x) = \max(0, x)$ (ADAVANNE, 2020). More about activation functions will be discussed in Section 2.8.

The learning process of neural networks involves two critical phases: feedforward and backpropagation. During feedforward propagation, input data flows from the network's input layer to its output layer, with intermediary functions computed within

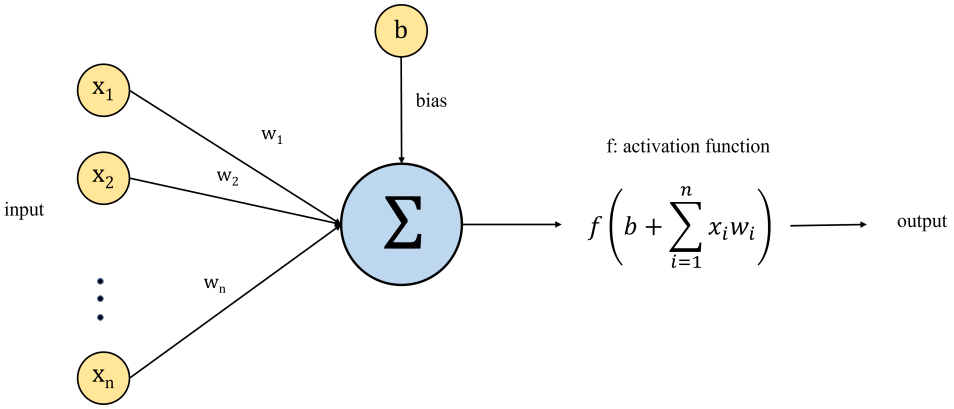


Figure 2.9 – The neuron consists of a linear operation from the inputs and a non-linear operation (activation function). Source: Author.

the hidden layers. Conversely, backpropagation aims to minimize the cost function (loss function) by iteratively adjusting the network's weights, reducing the disparity between the predicted output and the target output vector.

2.8 Activation functions

To effectively process a diverse amount of data and solve complex real-world problems, neural networks usually require the application of an activation function within their mathematical modeling. This function produces more dynamism in the network composition, which enables it to learn more complex and non-linear relationships within data from input to output. An activation function must be differentiable to allow for the implementation of backpropagation, an algorithm used to compute the errors or losses with respect to the weights in a neural network (see Section 2.10). This allows the iterative optimization of weights, utilizing techniques such as Gradient Descent to reduce errors (SHARMA *et al.*, 2020).

One of the most significant challenges activation functions face is the Vanishing Gradient problem. This issue occurs when gradient values become extremely small, or "vanish", during backpropagation, leading to minimal weight updates. As a result, the network struggles to learn effectively, with weights becoming saturated and progress stalling. Consequently, the loss ceases to decrease, preventing the network weights from being trained and appropriately updated (GUSTINELI, 2022). Saturated neurons are what the neurons with inadequately updated weights are called. Some of the main activation functions will be discussed next.

- **Sigmoid**

  The Sigmoid function is the most commonly used activation function due to its

non-linear nature. It transforms input values into a range between 0 and 1. This behavior is illustrated in Figure 2.10, and its mathematical definition is:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{2.3}$$

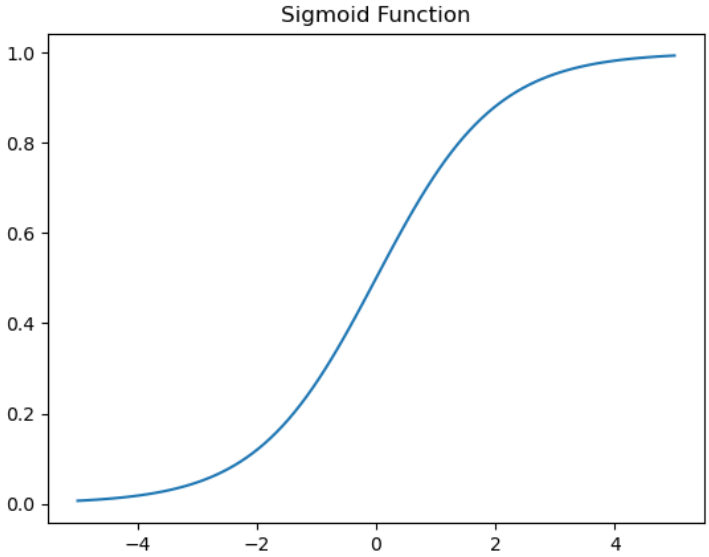


Figure 2.10 – Sigmoid Activation function. Source: Author.

- **Tanh**

  The Hyperbolic Tangent function, Tanh, has a similar shape to the sigmoid function. However, it exhibits symmetry around the origin. This activation function leads to higher gradients than the sigmoid function, and its output values lies between -1 and 1. Its behavior is illustrated in Figure 2.11, and its mathematical definition is:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.4}$$

- **Softmax**

  The Softmax function can be seen as a generalization of the sigmoid function for multiclass classification. While the sigmoid function outputs values between 0 and 1, representing probabilities for individual classes in binary classification, the softmax function extends this capability to handle multiple classes. Unlike sigmoid functions, softmax outputs a probability distribution across all classes, with the sum of the probabilities equal to 1. This behavior is illustrated in Figure 2.12, and its mathematical definition is:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \tag{2.5}$$

  where N is the number of classes.

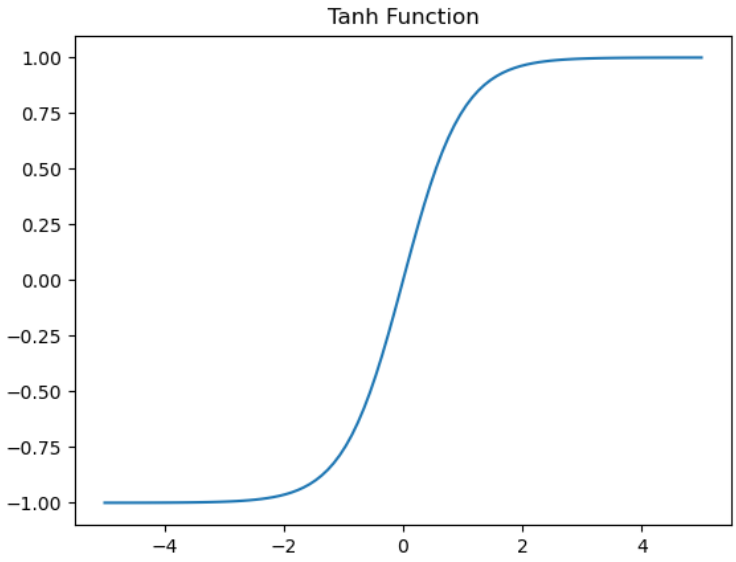


Figure 2.11 – Tanh Activation function. Source: Author.

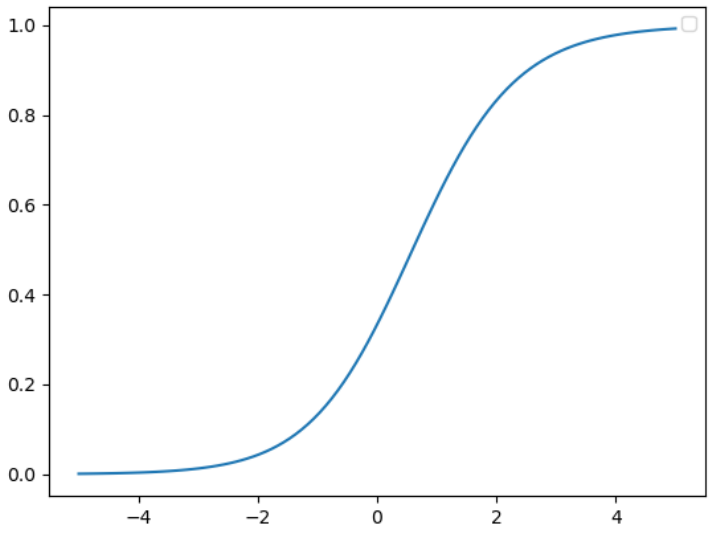


Figure 2.12 – Softmax Activation function. Source: Author.

- **ReLU**

  ReLU (Rectified Linear Unit) is a widely used non-linear activation function in neural networks. Its advantage is that it selectively activates neurons, ensuring that not all neurons are active at the same time. Specifically, a neuron is deactivated when the output of the linear transformation is less than zero. This behavior is depicted in Figure 2.13, and its mathematical definition is:

$$\text{ReLU}(x) = \max(0, x). \tag{2.6}$$

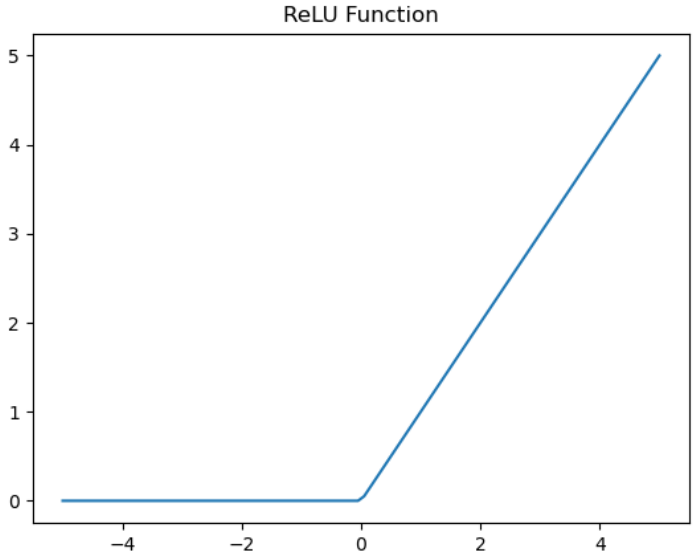


Figure 2.13 – ReLU Activation function. Source: Author.

2.9 Loss functions

A loss function quantifies how well the model fits the training data by evaluating the mathematical difference (error) between predicted and actual output values during the neural network training. In supervised learning, the employed loss function depends on the type of the problem, whether it is regression or classification. The primary loss functions in regression models are Mean Squared Error (MSE) and Mean Absolute Error (MAE). In classification tasks, the loss functions covered here are Binary Cross-Entropy, and Categorical Cross-Entropy (TERVEN *et al.*, 2023).

2.9.1 Binary Cross-Entropy Loss

Binary Cross-Entropy (BCE), or log loss, is employed in binary classification to assess the difference between the predicted class probability and the actual class label. It is often used to measure how different two probability distributions are. The loss function is defined as:

$$J = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)\right], \tag{2.7}$$

where p_i represents the predicted probability for the i-th sample, y_i denotes the true class of the i-th sample and N is the total number of samples.

2.9.2 Categorical Cross-Entropy Loss

Categorical Cross-Entropy Loss is a common choice when dealing with multiple class problems. It assesses the difference between the predicted probability distribution

and the actual distribution. The loss function is mathematically defined as:

$$J = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \cdot \log(p_{ij}), \tag{2.8}$$

where p_{ij} represents the predicted probability for class j of the i-th sample example, y_{ij} denotes if class j is the true class for the i-th sample example, N is the total number of examples and C is the total number of classes.

2.10 Gradient descent algorithm

The gradient descent algorithm is used during the training phase of the model to optimize the loss function employing iterative updates on the learning parameters w, known as weights. This procedure is what is referred to as the learning process. The gradient descent algorithm comprises three main stages:

1) **Feed-forward Propagation**: The input vector x undergoes a series of mathematical operations throughout the entire model to generate a prediction $\hat{y}$.

2) **Loss Function Computation**: After the feed-forward is complete, the loss function is calculated alongside the gradient of the loss function for each weight w. The gradient is computed to determine the correct direction towards a local minimum of the loss function, as represented in Figure 2.14.

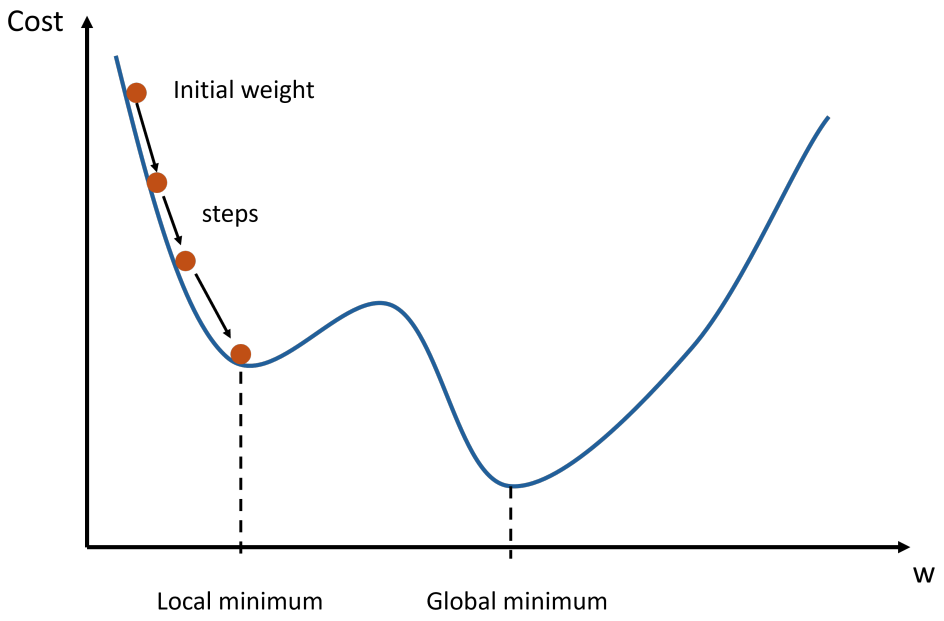


Figure 2.14 – The aim of the gradient descent is to find the minimum points of a cost function. Source: Author.

3) **Backpropagation**: The error is propagated back through the network, and the weights (w) associated with reducing the loss are updated. This backward propagation

aims to adjust the parameters to minimize the loss further.

These three steps are repeated multiple times during the training phase. So, the gradient descent operates as a method to minimize the loss function $J(\theta)$ that is dependent on the model's parameters $\theta \in \mathbb{R}^d$. This minimization process involves updating the parameters in the direction opposite to the gradient of the loss function, denoted as $\nabla_\theta J(\theta)$. The learning rate (α) controls the sizes of the steps towards the direction of a local minimum. Analogously, this approach involves going down the slope of the function's surface, similar to going downhill until reaching a valley (RUDER, 2016).

There are three approaches to gradient descent, which depend on the amount of data used to compute the gradient: batch gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent. The choice typically involves balancing the trade-off between the accuracy of parameter updates and the time required to perform these updates.

Batch gradient descent calculates the gradient of the cost function considering the learning rate (α) for the entire training dataset at once. Since it requires computing gradients for the entire dataset for each update, batch gradient descent can be slow and impractical for datasets too large to fit into computer memory. The formula of weights update for Batch gradient descent is:

$$\theta_{n+1} = \theta_n - \alpha \nabla_\theta J(\theta_n). \tag{2.9}$$

On the other hand, stochastic gradient descent (SGD) updates parameters for each training example individually. Unlike batch gradient descent, which recomputes gradients for similar examples before each update, SGD performs one update at a time. This approach is generally faster and better suited for online learning. The formula for updating weights in stochastic gradient descent is:

$$\theta_{n+1} = \theta_n - \alpha \nabla_\theta J(\theta_n; x^{(i)}; y^{(i)}). \tag{2.10}$$

Finally, the mini-batch gradient descent combines the advantages of both approaches by updating parameters for every mini-batch of n training examples. This method reduces the variance in parameter updates, leading to more stable convergence, and uses highly optimized matrix operations commonly found in cutting-edge deep learning libraries, making the gradient computation of mini-batch very efficient. The formula of weights update for mini-batch gradient descent is:

$$\theta_{n+1} = \theta_n - \alpha \nabla_\theta J(\theta_n; x^{(i:i+n)}; y^{(i:i+n)}). \tag{2.11}$$

However, there are variations of SGD. In the standard gradient descent algorithm, the learning rate (α) is fixed, which means it is often set high initially and then adjusted manually in steps. The Adam optimizer is an improvement over standard SGD, which provides adaptive learning rates for each parameter based on the history of gradients. This allows the optimizer to converge faster and more accurately (KINGMA; BA, 2017).

2.11 Regularization

According to Goodfellow *et al.* (2016), regularization is "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error". Several techniques have been proposed to improve the accuracy, generalization over unseen examples, and convergence speed for neural network training. These techniques are grouped under network regularization techniques, and they help avoid the overfitting problem discussed in Section 2.4 and improve the generalization of the trained networks. The main techniques will be covered next.

Dropout is an algorithm where, during training, hidden unit activations are randomly set to zero (dropped out) with a certain probability, typically between 0.1 and 0.25. This process helps prevent units from relying too much on specific input connections, reducing their co-adaptation. By randomly dropping units, dropout effectively mimics training multiple networks with varied initial parameters and averaging their outputs, promoting a more robust and generalized model (CAKIR, 2019).

Another regularization technique is the Batch Normalization (BN). It fundamentally aims to standardize the activations within each neural network layer, ensuring a Gaussian distribution with zero mean and unit variance. This process enhances network generalization and mitigates sensitivity to poor initialization. BN is implemented by integrating normalization layers following fully connected or convolutional layers preceding non-linear activation functions. As a differentiable operation, normalization preserves the training procedure. The trainable parameters (weights and biases) for normalization are learned iteratively, computing running averages specific to each mini-batch. It acts as a preprocessing step, preparing the data at each network layer (KOUTINI, 2018).

The following regularization technique is Early Stopping (ES). Early stopping involves splitting the available data into training, validation, and testing sets. The algorithm keeps track of the error on the validation set throughout the training process. Suppose the validation error does not decrease during a given number of iterations (known as "patience"). In that case, the training is halted, and the weights corresponding to the minimum validation error are retained (GENÇAY; QI, 2001). Figure 2.15 shows the in-

dicated point where an ES is executed.

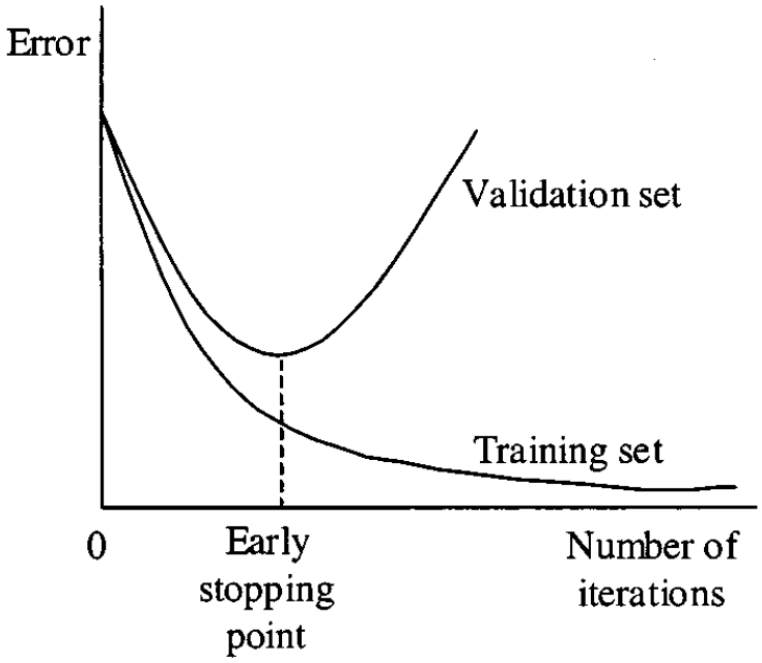


Figure 2.15 – Example of an Early Stopping application from Gençay and Qi (2001).

Lastly, the Data Augmentation technique employs various methods to create additional training samples from the original data by introducing random variations and adjustments while preserving the class labels. Data augmentation primarily aims to enhance the model's ability to generalize. By exposing the network to diverse, slightly altered versions of the input data, it learns more resilient and versatile features (ROSEBROCK, 2017a). More about this technique will be described in Section 2.18.

2.12 Convolutional Neural Networks (CNN)

Convolutional Neural Networks were first introduced to deal with image recognition patterns once they could learn the main spatial features presented in an array of pixels. These features can be edges, shapes, or structures commonly belonging to a particular class of images. In the case of acoustics, the image is often a spectrogram, Mel spectrogram, MFCC, or any other kind of temporal-frequency representation. CNNs differ from the fully connected layers because not all input neurons are connected to all output neurons, leading the model to learn local patterns. CNNs use small filters, called kernels, which slide over the input image or spectrogram and perform convolutions at each position. These filters contain the weights to be learned by the model during the training. Figure 2.16 shows a 2×2 kernel example that slides over the input image and results in an

output (called feature map). In this example, the 2×2 kernel is multiplied element-wise by a 2×2 region of the input (highlighted in blue), and the resulting values are summed.

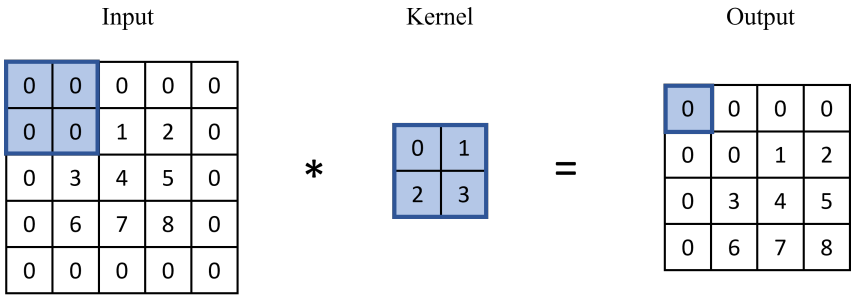


Figure 2.16 – Example of a convolutional layer. Source: Author.

CNNs exhibit two important properties. The first is local invariance, meaning a pattern can be recognized anywhere in the image. The second is the ability to learn spatial hierarchies, for example, allowing the network to learn edges from pixels, shapes from edges, and complex objects from shapes (ROSEBROCK, 2017b).

Convolutional layers are often followed by a pooling layer, which replaces the output at each location with a summary statistic of nearby outputs. The outputs of a pooling layer are usually the maximum value or the average in each region. Figure 2.17 demonstrates a Max-pooling layer with a filter size of 2×2.

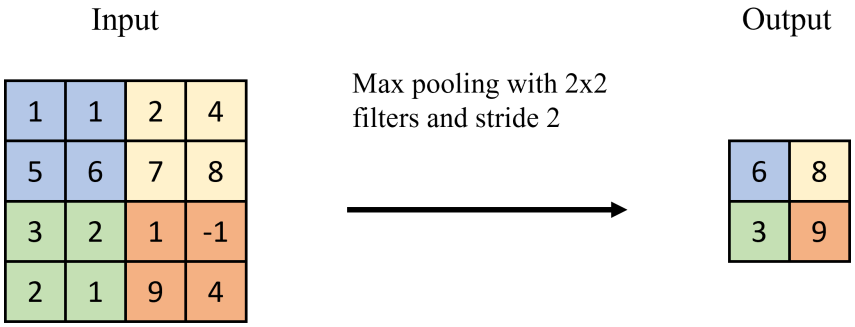


Figure 2.17 – Example of a Max-pooling layer carried out using a 2x2 filter (kernel) and stride of 2. Source: Author.

This reduces the image size, and it is useful when spectral pattern positions of a class present small shifts. Reducing the spatial size also minimizes the parameters and computational cost, hence controlling overfitting. After the convolutional layers, the outputs are often connected to fully connected layers where the prediction is made. CNN architectures are currently part of the state-of-the-art in the field of sound classification (ROCH, 2021).

2.13 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN) identify patterns in sequential data such as handwriting, genomes, text, or time series, often encountered in industrial settings, e.g., stock markets or sensors. They can also apply to images if these are decomposed into patches and treated as a sequence. Unlike Feedforward Neural Networks, where information passes through the network without cycles, an RNN transmits information back into itself. This enables RNNs to extend the functionality of Feedforward Networks by considering not only the current input (x_t) but also previous inputs ($x_{0:t-1}$) (SCHMIDT, 2019). RNNs have various applications, including language modeling and text generation, speech recognition, image description generation, and video tagging on a higher level.

At each time step t (also known as a frame), this recurrent neuron receives inputs $x(t)$ as well as its output from the previous time step $y(t-1)$, as shown in Figure 2.18.

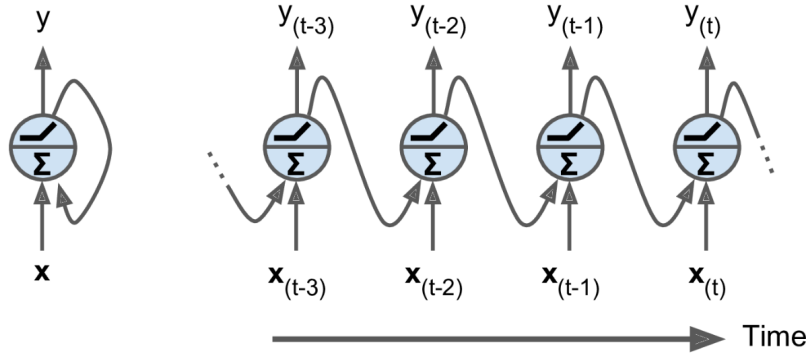


Figure 2.18 – A recurrent neuron (left), unrolled through time (right) (SCHMIDT, 2019).

Each recurrent neuron possesses two sets of weights: one for the inputs $x(t)$ denoted as weight vector w_x, and the other for the outputs of the previous time step, $y(t-1)$, denoted as weight vector w_y. Considering the entire recurrent layer rather than just one neuron, we can arrange all weight vectors into two weight matrices, W_x and W_y. The output vector of the entire recurrent layer can then be computed as:

$$y(t) = f\left(x(t)^T \cdot W_x + y(t-1)^T \cdot W_y + b\right), \tag{2.12}$$

where b is the bias term and $f(.)$ is the activation function.

Similar to many neural networks, RNNs also face issues of vanishing or exploding gradients. This issue was the reason for the development of Long Short-Term Memory units (LSTM). As a variation of RNN, LSTM has shown superior performance over traditional RNNs across various tasks.

The LSTM cell has the capacity to make decisions about what information to keep in the long-term state (through the input gate), what to forget (using the forget gate), and what information to read and output in the current time frame (by the output gate). Due to this mechanism, important information data are preserved from being disrupted by less relevant memory contents from more recent data points.

Additionally, there are bidirectional RNN models that incorporate information from both past and future time steps. These models utilize data from previous and later time frames, allowing a more comprehensive understanding of temporal patterns in sequences.

2.14 Performance Metrics

To assess the effectiveness of a sound classification model, CAKIR (2019) suggests comparing the model's binary predictions against the true labels in the test or evaluation dataset. This comparison involves calculating various performance metrics based on these binary outputs. Commonly utilized performance metrics for this task include accuracy, precision, recall (also called true positive rate), F1-score, error rate, and the area under the Receiver Operating Characteristic (ROC) curve.

The performance metrics for sound classification use a standard set of intermediate statistics. The final performance is derived from these metrics and intermediate statistics. These statistics encompass the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Some performance metrics can be computed individually for each class based on these statistics, while others can be calculated globally by aggregating the statistics across all classes.

Accuracy assesses the percentage of accurately classified instances from the total number of objects in the dataset. To calculate this metric, the number of correct predictions is divided by the total number of predictions generated by the model, as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{2.13}$$

Precision is a metric that reveals the proportion of correctly identified instances among all the detections made by the system for a specific class. It is computed as:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.14}$$

On the other hand, recall or true positive rate (TPR) indicates the ratio of correctly detected instances among all the instances of a specific class. Its calculation is

defined as:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.15}$$

The F1-Score is the harmonic mean of precision and recall, and it is calculated using the formula:

$$F1 = \frac{2 \cdot P \cdot R}{P + R}. \tag{2.16}$$

The ROC curve differs from other performance metrics in that it doesn't rely on a single threshold for converting detection probabilities into binary outcomes. Instead, the ROC curve is generated by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) across a range of threshold values. TPR is calculated equally to recall, and FPR is calculated as:

$$FPR = \frac{FP}{FP + TN}. \tag{2.17}$$

The area under the curve (AUC) from the ROC curve also indicates the system performance, as a high-performance method should have a greater TPR than FPR at each threshold value.

2.15 Cross-validation

Cross-validation is a model evaluation technique to assess a machine learning model's generalization performance capacity. It offers more stability and comprehensiveness than a traditional single split into training and test sets since it involves splitting the data into multiple subsets (folds) with subsequent training of models. An important cross-validation method is the k-fold cross-validation, where 'k' denotes a user-defined parameter typically set to 3, 5, or 10. In the case of 5-fold cross-validation, the dataset is initially divided into five approximately equal-sized partitions, known as folds. After that, a series of models are trained. Initially, the first model is trained employing the first fold as the test set, while the four remaining folds (2-5) are employed as the training set. Subsequently, the model is constructed using data from folds 1, 3, 4, and 5, with the accuracy evaluated on fold 2. So, this process is carried out five times, in the example, with each model trained utilizing a different fold as the test set and the remaining folds as the training set. This procedure is exemplified in Figure 2.19. The accuracy is computed for each one of the five splits of the data, facilitating a comprehensive evaluation of the model's performance (PAJANKAR; JOSHI, 2022). Finally, an average accuracy can be computed through all the splits.

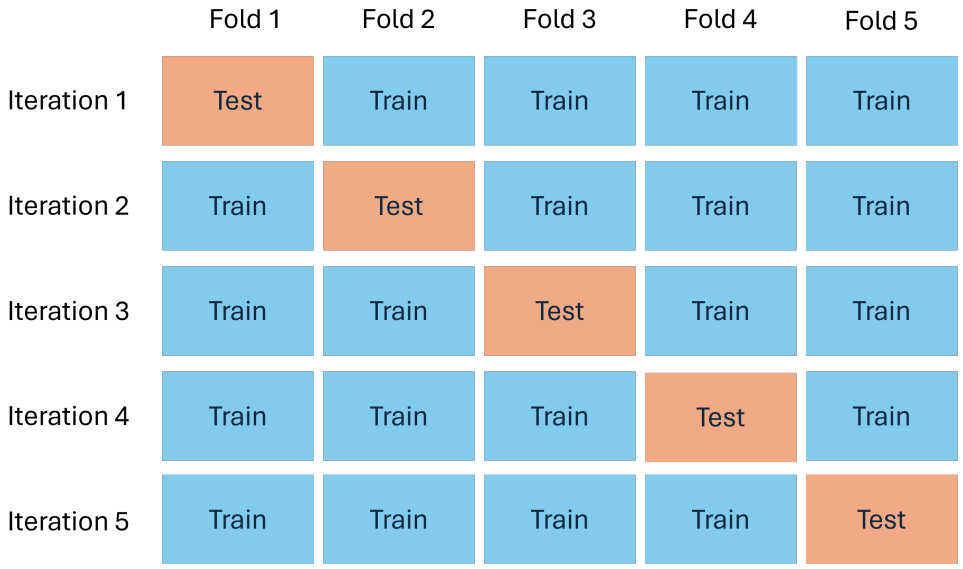


Figure 2.19 – Schematic example of a 5-fold cross-validation. Source: Author.

2.16 Sound Event Detection and Sound Classification

The task of sound event detection (SED) involves identifying a predominant event amidst background sounds. This objective includes detecting the start and conclusion of the sound event, as well as determining its type. Identifying the type of event within a sound segment is referred to as sound classification (NASIRI, 2021), and also named Environmental Sound classification (ESC) when dealing with non-speech signals. The SED and ESC representation is seen in Figure 2.20.

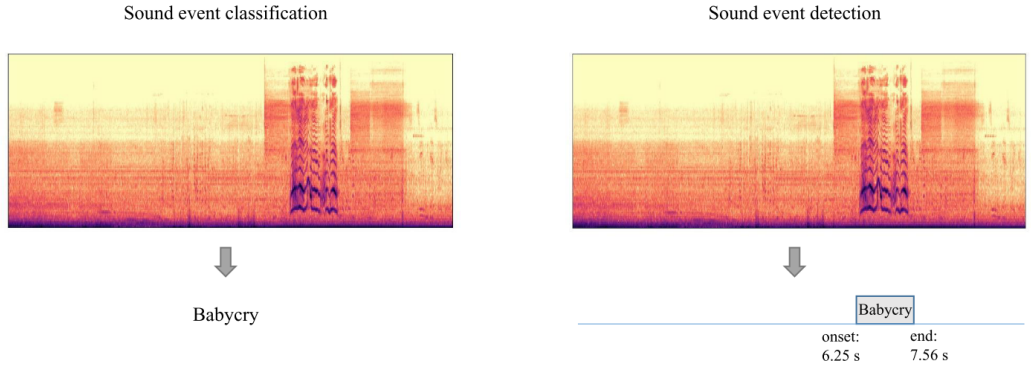


Figure 2.20 – SED and ESC difference represented in examples (NASIRI, 2021).

Urban environments present complex soundscapes where multiple sound events often co-occur. Consequently, noise monitoring systems in cities must typically address the challenge of multi-label audio classification. This characteristic of urban sound, where multiple sound events overlap, is known as polyphonic audio (CAKIR *et al.*, 2015). Figure 2.21 illustrates this concept. In this context, the algorithm continuously classifies fixed-duration audio segments, represented as time frames (denoted as t).

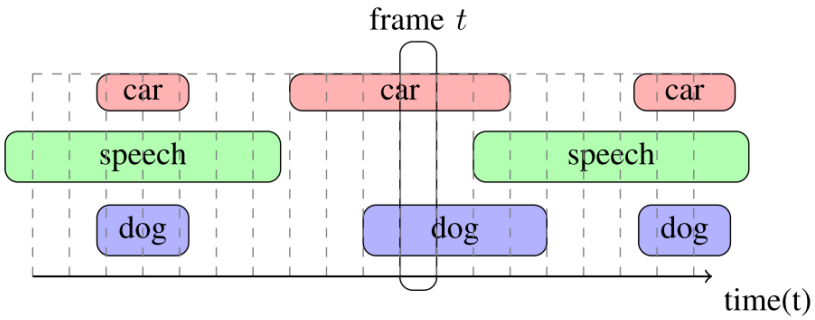


Figure 2.21 – Polyphonic SED example from Cakir *et al.* (2015).

2.17 Mel Scale and Mel Spectrogram

The human hearing perception is non-linear across the spectrum. The Mel scale is a psychoacoustic scale that reflects key aspects of human hearing: humans perceive pitch as linearly related to frequency in the lower frequency range. However, at higher frequencies, the relationship between perceived pitch and actual frequency becomes more logarithmic (GALILEU, 2020). Figure 2.22 illustrates this non-linear relationship graphically.

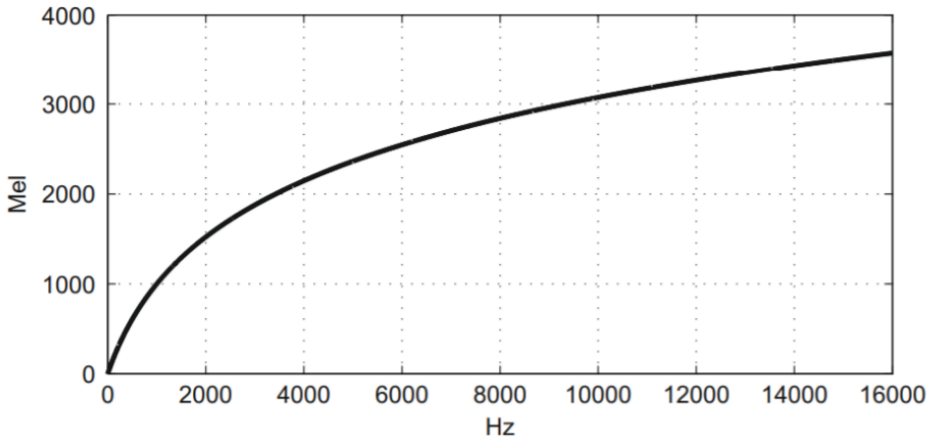


Figure 2.22 – Mel versus frequency scale relation from Galileu (2020).

The formula to convert frequency (in Hz) to pitch (in Mel) is expressed as:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{2.18}$$

where m corresponds to the Mel scale and f is the frequency in Hertz. Therefore, in the Mel domain, the frequency scale is adjusted so that pitches are perceived as equally spaced.

In deep learning, the audio signal is often represented as time-frequency domain data, and the Short-Time Fourier Transform (STFT) is the most common representation. Alternatively, the Mel scale can be used to consider human frequency perception. As

shown in Figure 2.23, a set of Mel filterbanks can be applied to the signal, resulting in the Mel spectrogram. The Log-Mel-spectrogram can be calculated in three steps (ARIAS-VERGARA *et al.*, 2021):

1. the signal is framed into short-time windows;
2. a Hamming time window is applied to each frame to compute the STFT;
3. a set of triangular filters in the Mel scale is applied, and the logarithm of the result is computed to obtain the Mel spectrogram.

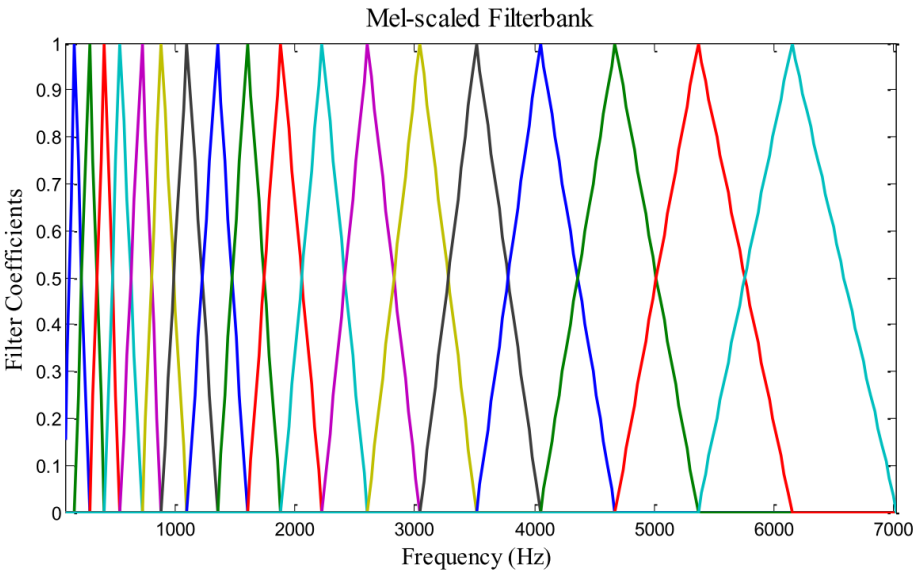


Figure 2.23 – 20 Mel filters bank from Yusnita *et al.* (2013).

2.18 Data augmentation for audio

Data augmentation is the process of applying one or more transformations to a set of annotated training samples to create new, additional training data. These transformations are made so that they do not change the semantic meaning of the labels (classes) and still represent their real behavior. In computer vision, for example, an image of a car can be rotated, translated, mirrored, or scaled and remains a valid image of a car. Therefore, these transformations can be used to generate additional training data while preserving the semantic content of the label. While training the network on this augmented data, the network can learn new features, become invariant to these transformations, and generalize better to unseen data (SALAMON; BELLO, 2017). Data augmentation can be applied to audio samples directly in the time domain or after the spectrogram has been extracted. Some of the most common techniques are:

- **Pitch shift**: It is a technique where the pitch of an audio is decreased or increased, randomly, by a factor, while the duration remains the same. Figure 2.24 shows an example of the resulting spectrogram using a pitch shift of 5 semitones in a voice audio.

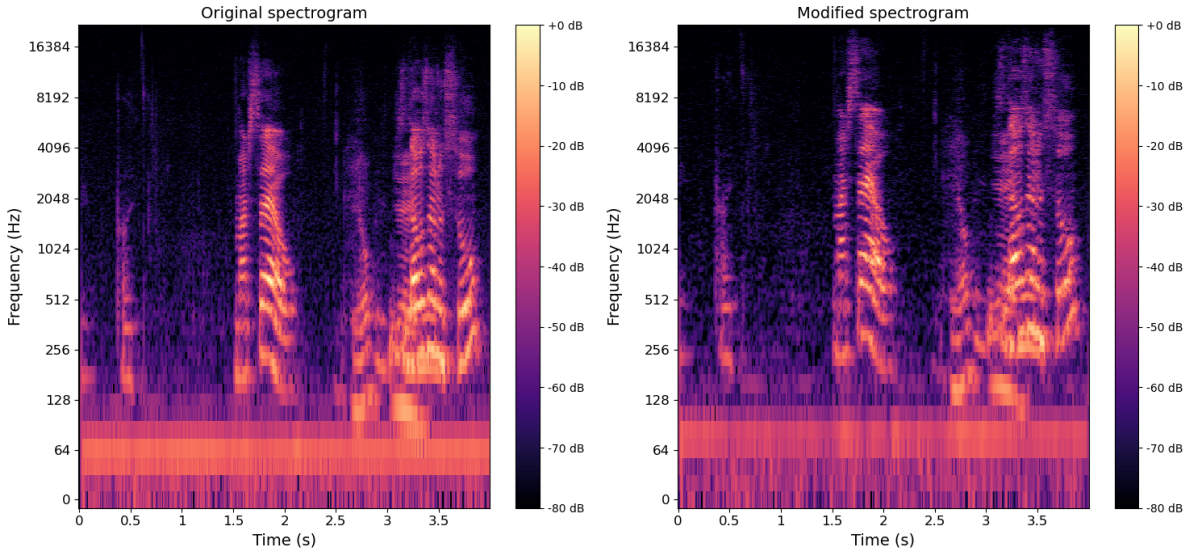


Figure 2.24 – Example of a spectrogram before and after the pitch shifting. Source: Author.

- **Time stretch**: This technique consists of slowing down or speeding up an audio duration by a ratio factor without changing the pitch. Figure 2.25 shows an example of the resulting time signal using a time stretch of 1.3 in a voice audio.

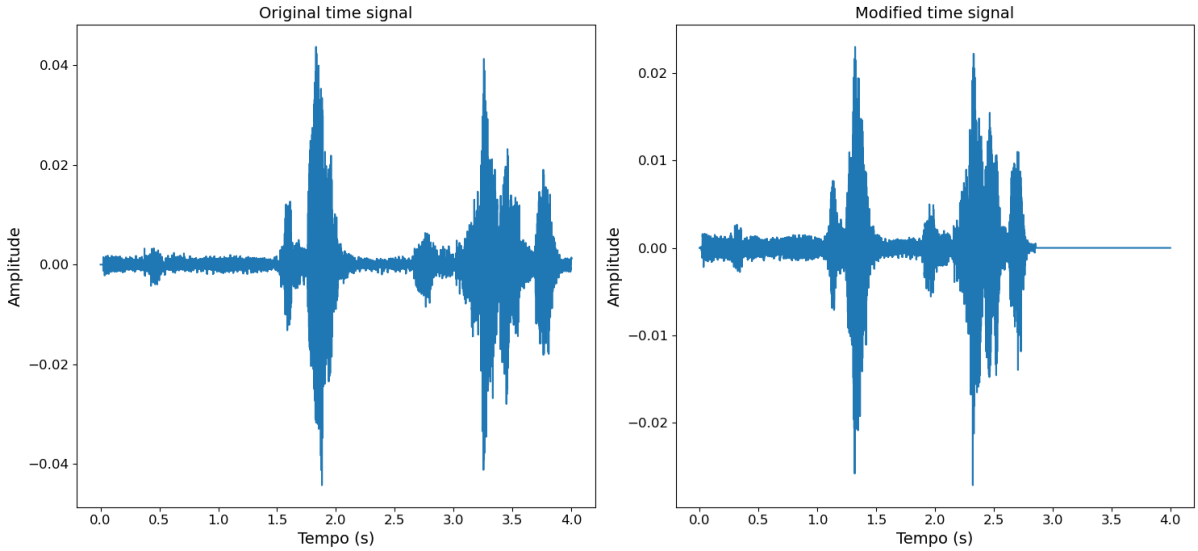


Figure 2.25 – Example of a time signal before and after the time stretch. Source: Author.

- **Synthetic noise**: In this method, a synthetic noise, typically a white noise, is created and added to an existing audio sample. Figure 2.26 shows an example of the resulting spectrogram of a voice audio with added white noise.

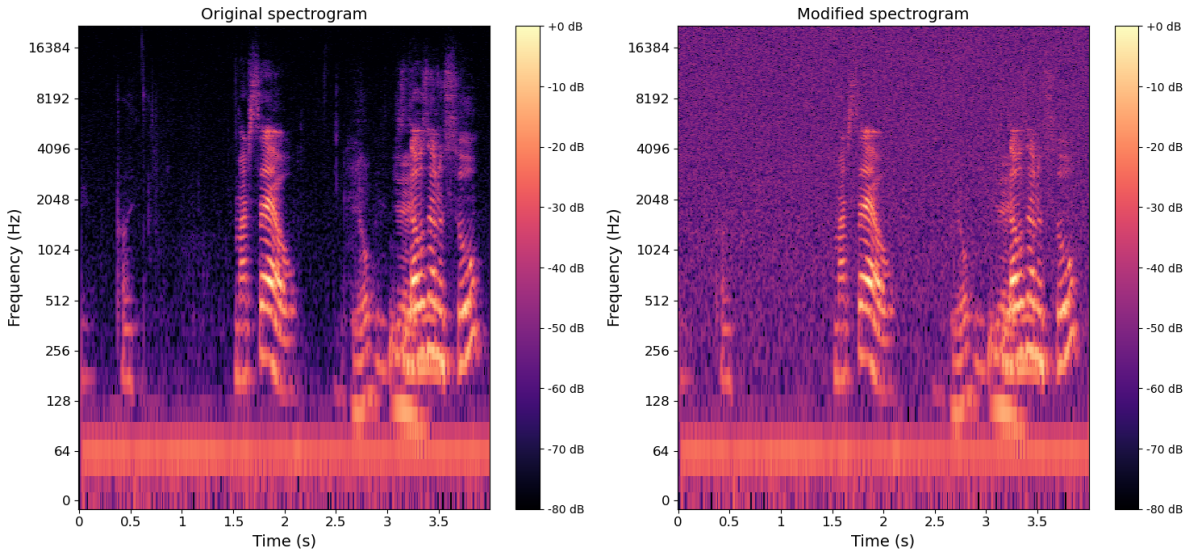


Figure 2.26 – Example of a spectrogram before and after the addition of synthetic white noise. Source: Author.

- **Time shifting**: This technique shifts the audio to the left or right, in the time domain, by a random factor. Figure 2.27 shows an example of the resulting time signal using a time shift of 0.5 in a voice audio.

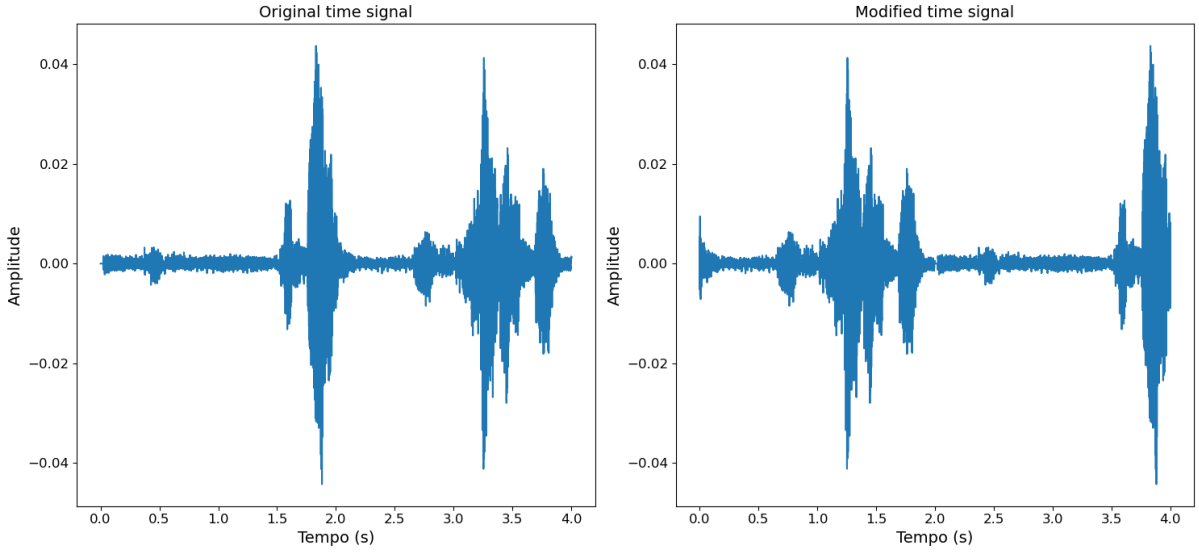


Figure 2.27 – Example of a time signal before and after the time shift. Source: Author.

- **Loudness**: The volume of all samples is increased or decreased at a random or fixed rate.

3 Methodology

This chapter first provides an overview of the state-of-the-art in urban sound datasets. It then describes the methodology used to create a new dataset of urban sounds for the city of São Paulo. Next, the chapter discusses the dataset preprocessing configuration employed in this work. Finally, it explains the machine learning and neural network architecture utilized to address the problem of sound classification on the new dataset.

3.1 Audio datasets for urban sounds

In this section, an overview of the current state of the main urban sound datasets is provided. The Sounds of New York City (SONYC) project (CARTWRIGHT *et al.*, 2019) is a research initiative focused on employing data-driven methodologies to address urban noise pollution in New York City. The project's primary objective is to study and comprehend the spatial and temporal behavior of noise pollution across the city. The SONYC project takes a real-time approach, leveraging years of measurement data. To support this research, the SONYC team has developed a specialized acoustic sensor designed for effective noise pollution monitoring. This sensor features high-quality sound acquisition capabilities while maintaining cost-effectiveness (MYDLARZ *et al.*, 2019).

The project involves deploying 55 low-cost acoustic sensor nodes across New York City to collect continuous real-time urban noise data. The research has gathered approximately 75 years of sound pressure level (SPL) data and 35 years of raw audio recorded from this sensor network. An extensive dataset allows for identifying noise patterns and detecting frequently overlooked sources of noise pollution in urban areas.

To deal with such a large amount of data, volunteers on the ZOONIVERSE citizen science platform assisted in the task of labeling the presence of 23 fine-grained classes of sounds, which were chosen consulting the New York City Department of Environmental Protection and respecting the New York City noise code. These 23 fine-grained classes can be grouped into eight coarse-grained categories: engine noise, machinery impact, non-machinery impact, powered-saw noise, alert signals, music, human voices, and dog sounds. The final dataset comprises 18,510 audio samples.

The sensor network does not only perform continuous SPL measurements but also records 10-second audio segments at random intervals during specified time frames. This data collection strategy is designed to facilitate the training and evaluation of machine listening algorithms deployed in acoustic sensors. During the measurement, the sen-

sor nodes constantly communicate with the server via a Virtual Private Network (VPN), and audio and SPL data are uploaded at a 1-minute interval.

Despite being the most complete available dataset, it still needs essential natural sounds, such as rain noise, birds, and insects. Additionally, the means of transportation need to be separated, as there are only generic labels for small, medium, and large-sized engines.

Another important dataset in urban sound field is the UrbanSound8K (SALAMON *et al.*, 2014). This dataset is commonly used as a benchmark test of urban sound classification models, as it comprises 8,732 audio files categorized into ten classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music. With the exception of "children playing" and "gunshot", which were added for variety, all other classes were selected based on an analysis of noise complaints collected through New York City's 311 services since 2010, in a total of over 370,000 complaints.

Given the manual effort required for audio annotation, the number of classes in the dataset was limited to 10 as a practical starting point. This decision was informed by a previous study, where Chu *et al.* (2009) conducted a listening test and found that subjects could identify environmental sounds with 82% accuracy within four seconds of the audio presentation. Consequently, a 4-second clip duration was adopted in the construction of this dataset. Furthermore, each sound occurrence was also described for its salience, which indicates whether a noise source was subjectively perceived to be in the foreground or background of the recording.

Despite its popularity for urban sound classification, UrbanSound8K only presents ten classes of sounds, which is generally insufficient for most urban sound monitoring cases. Additionally, they do not comprise specific classes for vehicles, which are the primary sound sources of metropolitan areas.

The IDMT-Traffic Dataset (ABESSER *et al.*, 2021) is designed to serve as a public evaluation benchmark for detecting and classifying passing vehicles on urban and rural roads. It comprises time-synchronized stereo audio recordings captured using both high-quality sE8 microphones and more affordable microelectro-mechanical systems (MEMS) microphones from four distinct recording locations, including three urban traffic sites and one rural road location in and around Ilmenau, Germany. The recording scenarios cover a few speed limits (30, 50, and 70 km/h) as well as some different road conditions, such as wet and dry surfaces.

In the dataset, four vehicle classes are included and distributed as 3903 events of cars, 511 events of trucks, 53 events of buses, and 251 events of motorcycles. This dis-

tribution reflects the natural imbalance of vehicle types typically encountered in everyday traffic scenarios. However, heavy vehicles often do not accurately represent the Brazilian vehicle fleet in most cases, as the Brazilian fleet tends to be older than the German fleet.

The ESC-50 dataset (PICZAK, 2015b) is another dataset commonly used as a benchmark for urban sound classification experiments. It comprises 2000 labeled environmental recordings, evenly distributed among 50 classes with 40 clips per class. These classes are grouped into five broad categories, each containing ten classes: animal sounds, natural soundscapes and water sounds, human (non-speech) sounds, interior/domestic sounds, and exterior/urban noises.

This dataset is organized into three main sets. The first is the primary labeled set, which includes 50 classes of diverse environmental sound sources. The second is a smaller proof-of-concept subset of 10 classes selected from the main dataset, serving as a simplified benchmark, and finally, a supplementary dataset of unlabeled audios suitable for unsupervised learning experiments. All datasets consist of sound clips collected from publicly available recordings from the Freesound project. The selection of classes of the labeled dataset was made in order to maintain a balance among major types of sound events while considering the limitations in the quantity and diversity of available source recordings, as well as considering how useful and distinctive they are.

To ensure data accuracy, each audio was individually evaluated and verified by the author through the annotation of segments containing events belonging to the given class. These annotations were then utilized to extract 5-second-long recordings of audio events, with shorter events padded with silence when necessary. The extracted samples were converted to a single channel and a sample rate of 44.1 kHz. The labeled datasets were then organized into five uniformly sized cross-validation folds.

Nevertheless, the dataset only contains 40 audio samples per class which is typically insufficient and challenging for many supervised learning applications, furthermore, most classes in the dataset are not particularly relevant for use in urban environments.

The ESC-10 dataset is a subset derived from the larger ESC-50 dataset. It comprises 10 classes extracted from the original, representing three broader groups of sounds: 1) transient or percussive sounds, characterized by distinctive temporal patterns, such as sneezing, dog barking, and clock ticking; 2) sound events with prominent harmonic content, exemplified by sounds like crying babies and crowing roosters; and 3) more structured noises or soundscapes, including rain, sea waves, fire crackling, helicopters, and chainsaws. This subset was designed to provide a more introductory and manageable problem for research purposes, compared to the full ESC-50 dataset.

The final important dataset covered in this review is the BCNDataset (VIDAÑA-

VILA *et al.*, 2020). This dataset comprises 363 minutes and 53 seconds of real-world audio recordings captured in the city center of Barcelona, recorded over two Saturdays from 22:00 to 03:00. It encompasses 14 distinct types of events, totaling 6076 event occurrences. These events were categorized into two main groups: "leisure" and "traffic", excluding rare events where the sound sources could not be determined or were a combination of sounds. The dominant type of noise event identified in the dataset is attributed to people, followed by road traffic noise, brakes, doors, and other rare occurrences.

3.2 The audio dataset

The dataset of this work was conceived to deal with the most common sound sources in the city of São Paulo, Brazil. It encompasses not only sound sources capable of producing high noise levels, such as construction noise and vehicles, but also considers sound sources that are generally neglected and capable of exceeding lower noise limits of residential zones at night, for example, birds, insects, and rain. Including a "Background noise" class is important regarding environmental sound classification since many regulations consider noise limits below 60 dB (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT), 2019) for comfort. The sounds are often close to the background noise level at such low sound pressure levels. It is important to note that the dataset only contains external urban sounds since internal environments comprise many different sound sources that are out of the scope of the project.

The dataset comprises five coarse classes: Construction noise, Animals, Alert signals, Vehicles, and Others. And a total of 19 fine classes: Impact Equipment, Non-impact equipment, Dog, Bird, Insect, Alarm, Siren, Car (pass-by), Motorcycle (pass-by), Heavy vehicle (pass-by), Heavy vehicle (idling), Helicopter, Airplane, Train, Music, Human, voice, Rain, Air conditioner and Background noise. The schematic taxonomy can be seen in Figure 3.1.

The labels colored in light green indicate that the samples from that sound source were collected from the UrbanSound8K dataset. In contrast, the labels in light orange signify that collecting enough audio files for these classes was not possible, and they will be excluded from this study. Future efforts will be undertaken in subsequent research to complete the entire dataset. Studies utilizing this dataset have been published using earlier and smaller iterations of it (BORIN *et al.*, 2023).

The "Air conditioning" class covers sounds generated by machinery like condensers, chillers, and similar equipment commonly found in air conditioning systems. This category includes mechanical noises associated with the operation of such units, like compressor hums, fan noises, and refrigerant flow sounds. The "Impact equipment" mainly

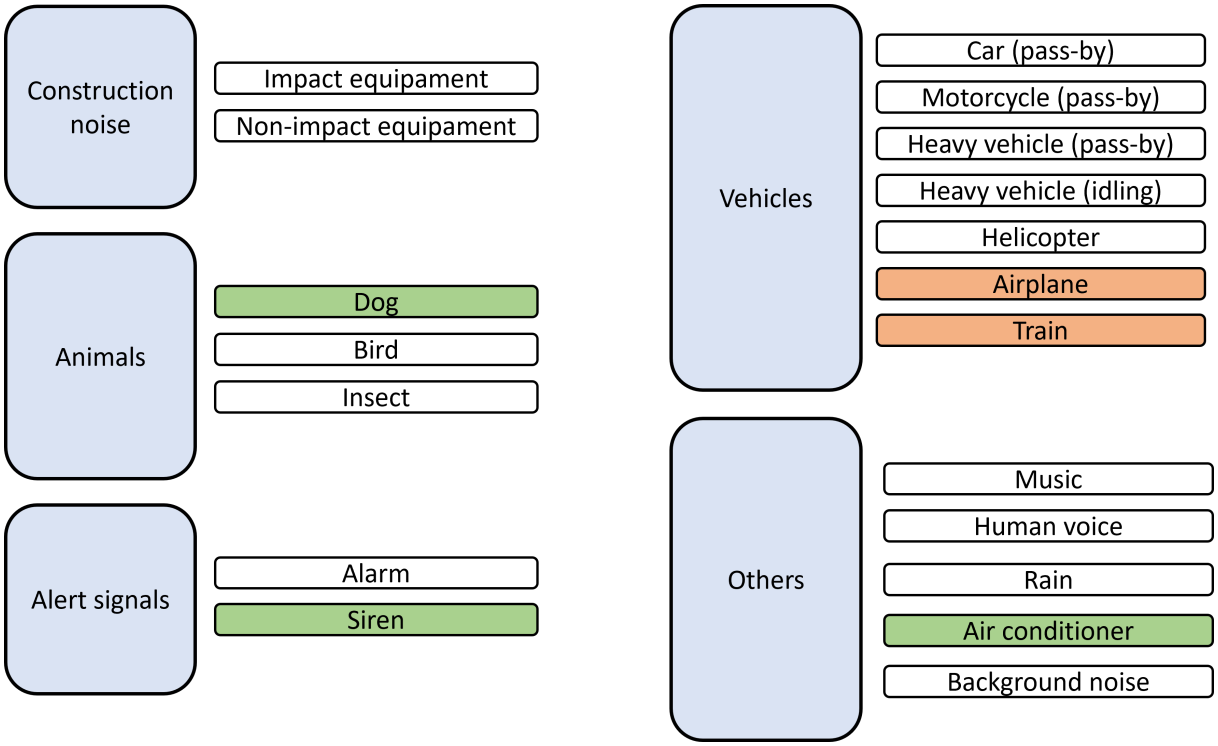


Figure 3.1 – Proposed dataset taxonomy. The light green indicates classes collected from UrbanSound8K dataset, while the light orange indicates classes which number of samples was not enough for this work. Source: Author.

comprises hammering and pile driver sounds, while the "Non-impact equipment" comprises sound samples such as drilling, sawing, sanding, power tools, and concrete mixers.

The audios were mainly collected in the city of São Paulo, Brazil, using a Tascam DR-05X recorder set to mono mode, sampling frequency of 44.1 kHz, 16 bits of depth, and saved in ".wav" file format. This configuration was chosen since it covers the typical audible range of the spectrum (20 Hz to 20 kHz), and resampling can be applied if necessary.

Audio recordings of up to 5 minutes each were conducted at various representative sites in the city of São Paulo at different times of the day. The recording places mainly included different streets, parks, construction sites, and facade measurements. Figure 3.2 illustrates a typical setup for vehicle recordings. Additionally, to ensure accurate traffic identification, video footage was captured concurrently with the audio using a smartphone, aiding in the labeling process due to the potential for vehicle sounds to be misleading.

After each recording session, the audio files were analyzed using Reaper (COCKOS INCORPORATED,), a digital audio workstation (DAW). The sound sources were carefully identified by listening to the files. Subsequently, audio segments corresponding to the identified sound sources were manually trimmed to a maximum duration of four seconds.

Figure 3.2 – Tascam recording the traffic noise in São Paulo, Brazil. Source: Author.

These trimmed segments were then saved as ".wav" files to their designated paths in the database, categorized according to their respective classes.

As mentioned in Section 3.1, Chu *et al.* (2009) conducted a listening test, revealing that subjects could identify environmental sounds with 82% accuracy within a 4-second duration. This finding was relevant for their adoption of a 4-second clip duration in their experiments with automatic classification. In a related study, Salamon *et al.* (2014) investigated the impact of audio duration on classification accuracy. They created ten versions of the UrbanSound8K dataset, varying the maximum slice duration from 10 seconds to 1 second. Their analysis compared the performance of five different algorithms: decision tree (J48), k-NN (with k = 5), random forest (500 trees), support vector machine (SVM) (radial basis function kernel), and a baseline majority vote classifier (ZeroR). The findings revealed consistent behavior across all classifiers: performance remained stable from 10 to 6 seconds, with a gradual decline after that. However, focusing on the top-performing classifier (SVM), no statistically significant difference was observed between 6-second and 4-second slices. This finding aligns with the results of Chu *et al.*, supporting the choice of 4-second slices for the UrbanSound8K dataset. Following these findings, a 4-second duration was also chosen as the maximum occurrence duration for the dataset in this work.

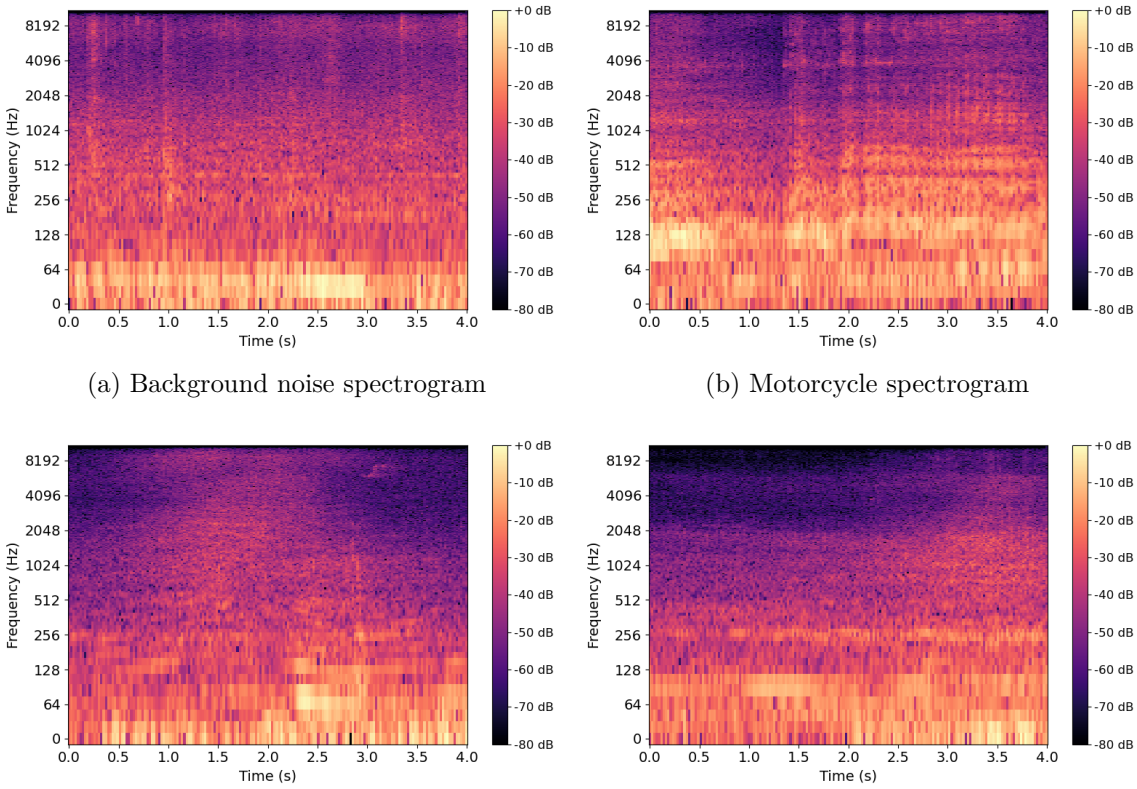
Table 3.1 presents the complete dataset used in this project, showing the class and its respective number of files. In this work, a total of 5864 audio files were used.

An example spectrogram of each class can be seen in Figures 3.3, 3.4 and 3.5. The Short-Time Fourier Transform (STFT) was computed using a hop size of 512 and a Hanning window of 1024 samples. The sampling rate was set to 22050 Hz. These spectrograms are only illustrative to comprehend differences between classes better. Section 3.3

Table 3.1 – Number of audio samples collected for each class.

Class	Number of samples
Background noise	413
Motorcycle	306
Car	339
Heavy vehicles (pass-by)	365
Heavy vehicles (idling)	315
Helicopter	320
Air conditioner	360
Construction noise (non-impact)	409
Construction noise (impact)	303
Rain	307
Human voice	398
Music	368
Dog	360
Insect	303
Bird	315
Alarm	320
Siren	360
Total	5864

will explore the actual data preprocessing configuration.



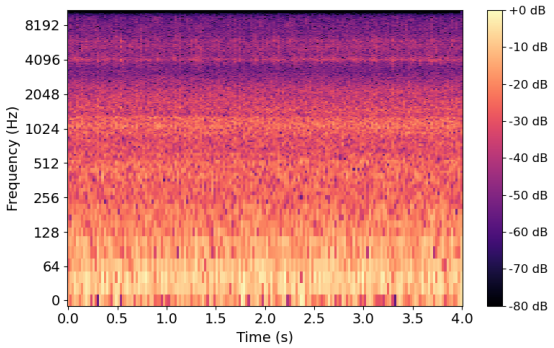
(a) Background noise spectrogram

(b) Motorcycle spectrogram

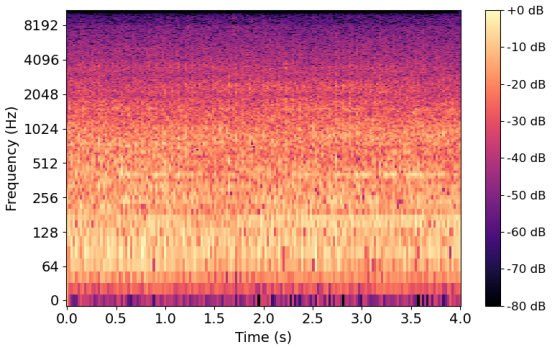
(c) Car spectrogram

(d) Heavy vehicle (pass-by) spectrogram

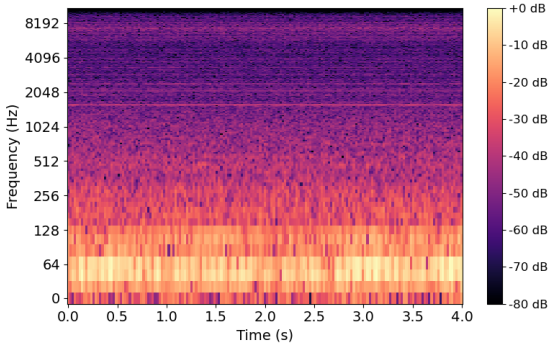
Figure 3.3 – Example spectrograms from classes: (a) Background noise, (b) Motorcycle, (c) Car, (d) Heavy vehicle (pass-by).



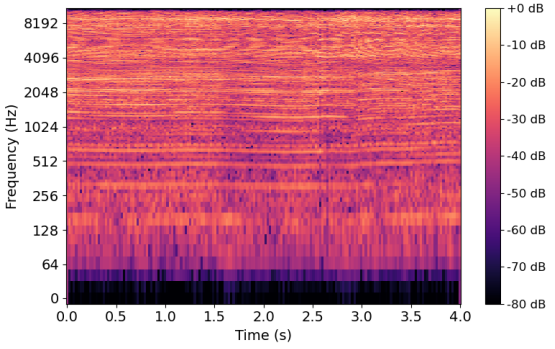
(a) Heavy vehicle (idling) spectrogram



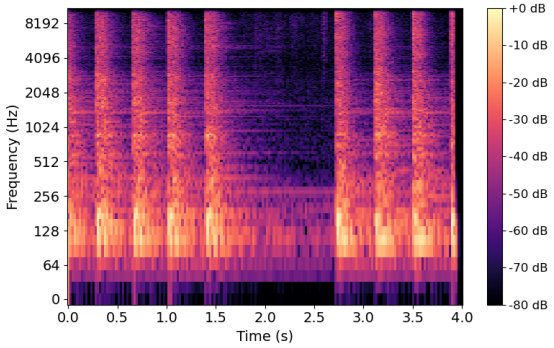
(b) Helicopter spectrogram



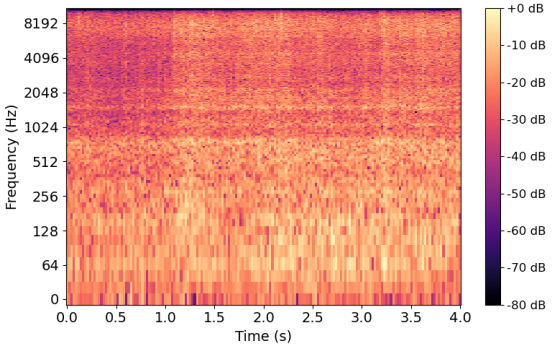
(c) Air conditioner spectrogram



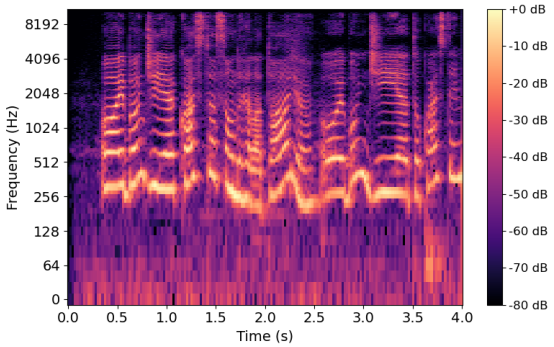
(d) Construction noise (non-impact) spectrogram



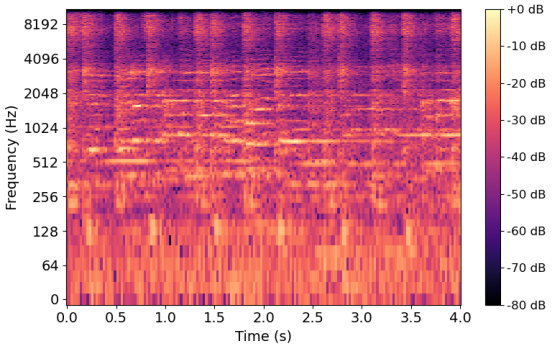
(e) Construction noise (impact) spectrogram



(f) Rain spectrogram

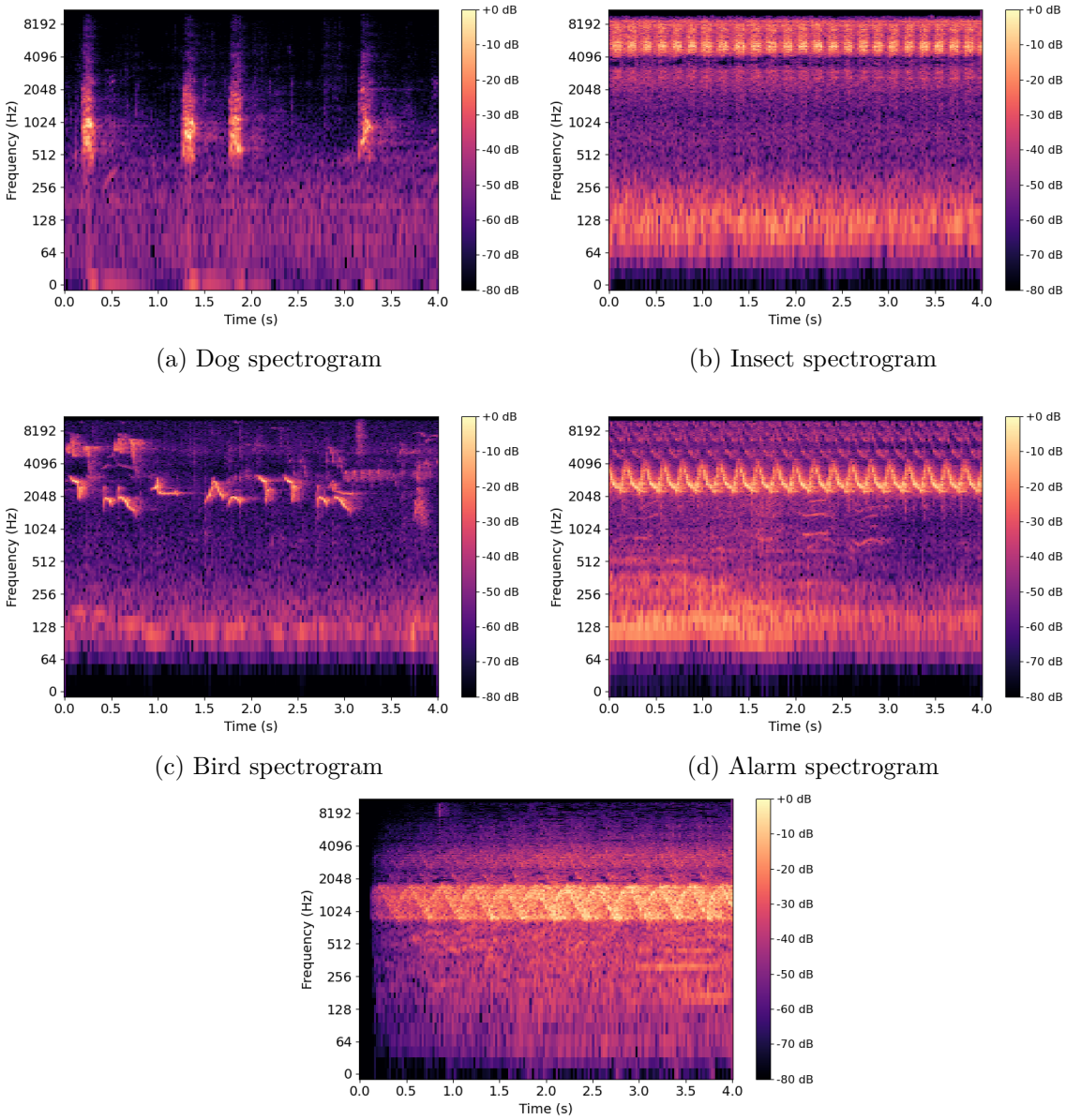


(g) Human voice spectrogram



(h) Music spectrogram

Figure 3.4 – Example spectrograms from classes: (a) Heavy vehicle (idling), (b) Helicopter, (c) Air conditioner, (d) Construction noise (non-impact), (e) Construction noise (impact), (f) Rain, (g) Human voice, (h) Music.



(a) Dog spectrogram

(b) Insect spectrogram

(c) Bird spectrogram

(d) Alarm spectrogram

(e) Siren spectrogram

Figure 3.5 – Example spectrograms from classes: (a) Dog, (b) Insect, (c) Bird, (d) Alarm, (e) Siren.

To perform a 5-fold cross-validation, the dataset was evenly divided into 5 folds, ensuring that each fold had the same class distribution, or as close as possible. Next section will cover data preprocessing.

3.3 Data Preprocessing

To satisfy the CNN model's requirement for fixed-length audio frames, all audio samples shorter than four seconds were padded with zeros to extend their length to four seconds.

A literature review was conducted to choose the most appropriate sample rate for this project. In studies from Boddapati *et al.* (2017) and Boddapati (2017), an accuracy comparison of sample rates (8 kHz, 16 kHz, and 32 kHz) applied to the ESC-10 and ESC-50 datasets revealed superior results with a sample rate of 16 kHz. Conversely, in the study from Högskola *et al.* (2018), no significant difference was observed among sample rates of 44.1 kHz, 32 kHz, and 16 kHz when using linear spectrograms as input. However, the utilization of Mel spectrograms indicated that a sample rate of 32 kHz yielded better results with the ESC-50 dataset. In investigations from Guo *et al.* (2022), a comparison of sample rates (8 kHz, 16 kHz, 44.1 kHz, and 48 kHz) across ESC-10, ESC-50, and UrbanSound8K datasets demonstrated that 44 kHz was consistently superior by 1 or 2 percentage points in most cases. Finally, in the study from Aljubayri (2023), an examination of sample rates (8 kHz, 16 kHz, and 44.1 kHz) using the UrbanSound8K dataset revealed that the optimal results were achieved with sample rates of 16 kHz and 44.1 kHz, depending on the input feature.

In general, sample rates ranging from 16 kHz to 44 kHz have been found to yield the best results for sound classification, with precision varying depending on input features. Therefore, a sample rate of 22050 Hz was selected for this study, considering both performance and memory consumption. For this reason, all audio files had to be resampled, and this task was conducted using the Librosa Python library (MCFEE *et al.*, 2021).

Next, all audio sample amplitudes were standardized to have a zero mean and a standard deviation of 1, resulting in an RMS of 1 and the same power level. This allows the model to ignore audio levels and focus on learning pattern features within the audio. The Short-Time Fourier Transform (STFT) was then calculated for each audio sample. The STFT used a Hanning time window with lengths of 8192 (372 ms), 4096 (186 ms), 2048 (93 ms), and 1024 (946 ms) samples. The respective hop sizes were 4096 (186 ms), 2048 (93 ms), 1024 (46 ms), and 512 (23 ms) samples. This process extracted the amplitude spectrogram, always ensuring a 50% overlap. Afterward, the spectrogram underwent processing through a Mel filter bank to divide the frequency spectrum into 16, 32, 64, and 128 Mel bands. The resulting spectrogram was then converted into a logarithmic scale to derive the Log-Mel spectrogram. Each audio sample thus results in a fixed frame according to the parameters used. A total of eight configurations were adopted in this work, with a summary of the parameters for all experiments provided in Table 3.2.

An example of the resulting Log-Mel spectrogram from a sample of human voice for each configuration is seen in Figures 3.6 and 3.7.

Table 3.2 – Summary of the 8 experiments configuration.

Configuration	Mel bands	Hop size	Window length	Mel spectrogram size
1	16	4096	8192	16×22
2	16	2048	4096	16×44
3	32	2048	4096	32×44
4	32	1024	2048	32×87
5	64	1024	2048	64×87
6	64	512	1024	64×173
7	128	1024	2048	128×87
8	128	512	1024	128×173

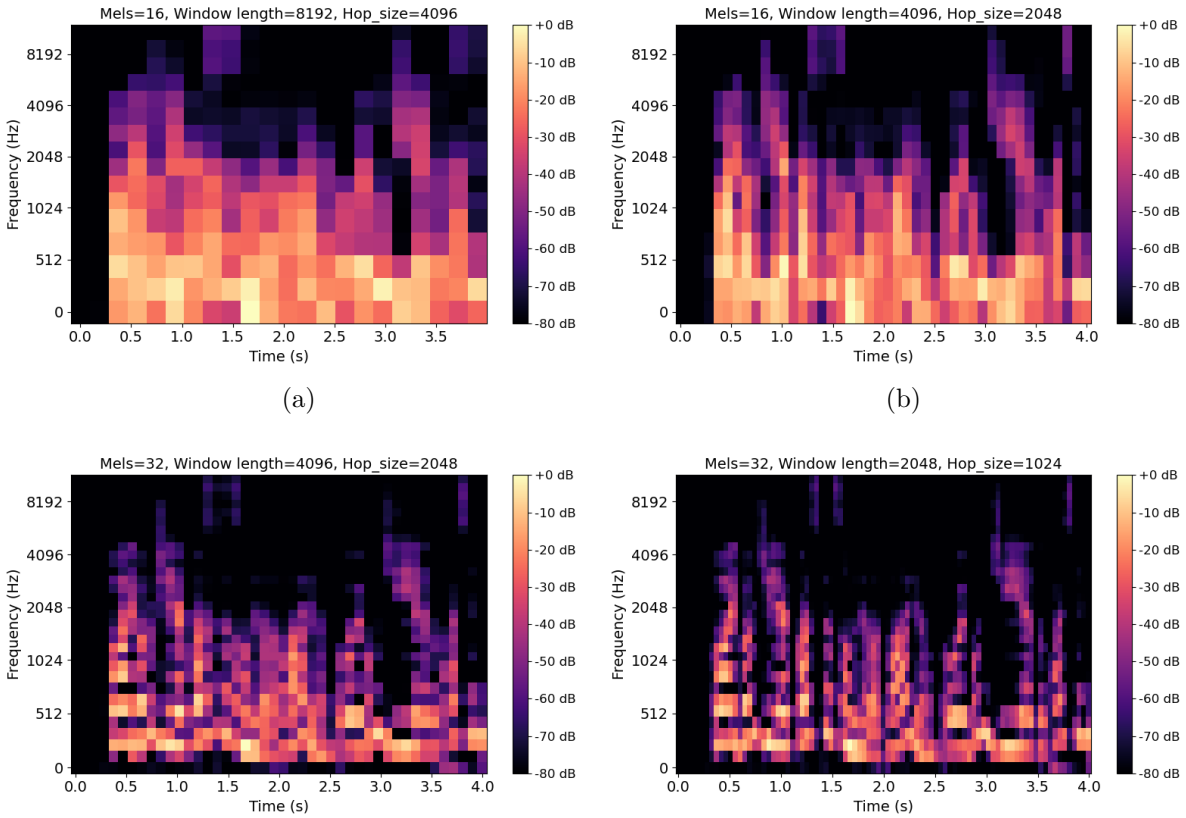


Figure 3.6 – Example spectrograms using (a) 16 Mel bands, hop size of 4096 and window length of 8192; (b) 16 Mel bands, hop size of 4096 and window length of 2048; (c) 32 Mel bands, hop size of 2048 and window length of 4096; (d) 32 Mel bands, hop size of 1024 and window length of 2048.

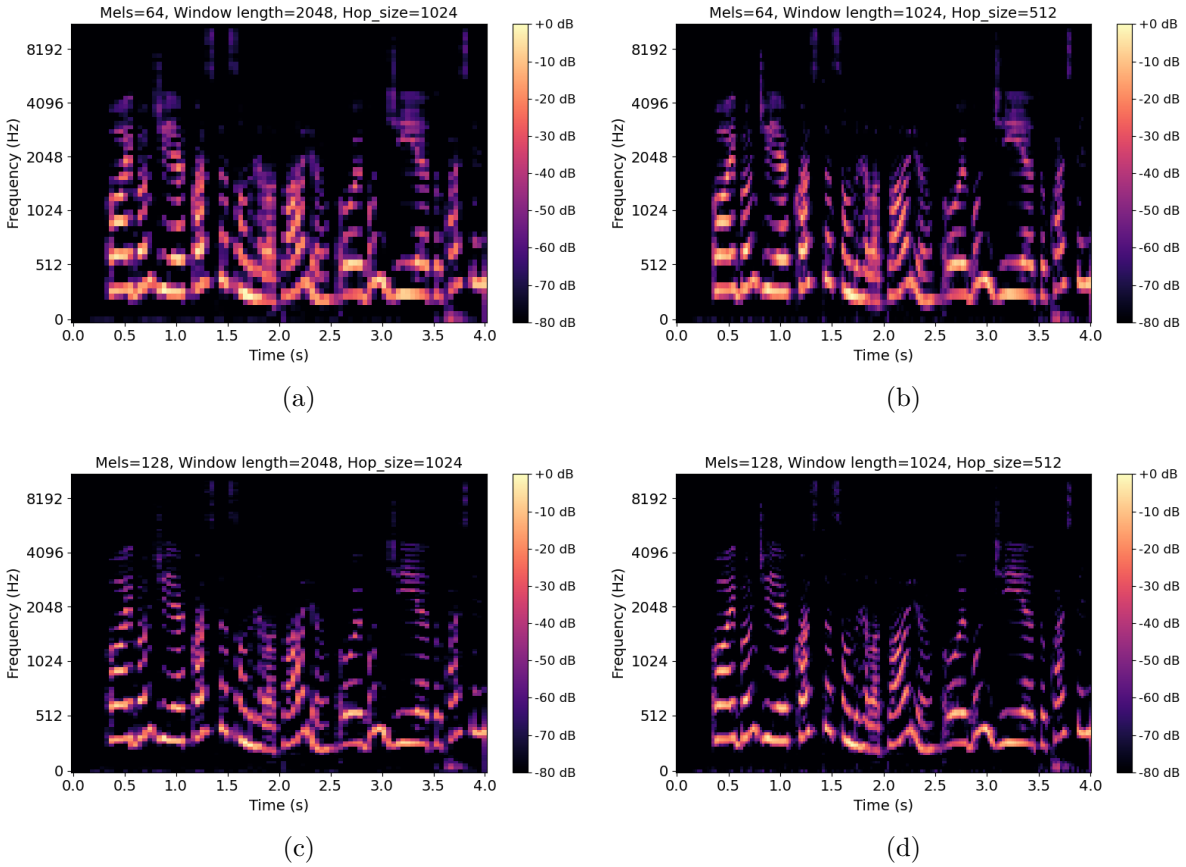


Figure 3.7 – Example spectrograms using (a) 64 Mel bands, hop size of 1024 and window length of 2048; (b) 64 Mel bands, hop size of 512 and window length of 1024; (c) 128 Mel bands, hop size of 1024 and window length of 2048; (d) 128 Mel bands, hop size of 512 and window length of 1024.

3.4 Machine Learning architectures

This section outlines the machine learning architectures used in this work. First, a simple linear support vector machine (SVM) model was selected as a baseline to compare with the subsequent, more complex models. This also allows for observing the behavior of a linear model in contrast to non-linear ones. Next, a CNN was chosen for its strong performance in classification tasks. A second deep learning model based on RNN (LSTM) was also used for comparison. Finally, a hybrid model was implemented, combining a CNN block and an RNN layer known as a CRNN (Convolutional Recurrent Neural Network).

3.4.1 Linear SVM

A linear SVM algorithm was first trained to solve the classification task to verify the necessity of implementing non-linear deep learning models. It served as a reference

to check the accuracy improvement of the more complex models. The kernel was linear, and the Regularisation Parameter (C) was set to 1. The model was configured to output probability estimates for each class.

3.4.2 CNN model

The first deep learning model covered in this work consists of a CNN model, which was chosen due to its good performance on audio classification tasks (ROCH, 2021). The CNN block includes three sets of alternating convolutional and max-pooling layers. Batch normalization is applied after each max-pooling operation. The CNN layers help reduce the data dimensions while extracting crucial features for classification. Then, the data is flattened and fed into a fully connected layer, followed by a dropout layer (0.3) before the output, as depicted in Figure 3.8, with the operations and its respective number of filters and kernel shapes. A Relu is used as an activation function for both the convolutional and dense layers, except for the last one where a Sigmoid function is applied. An L2 regularizer is used in the fully connected layer as well.

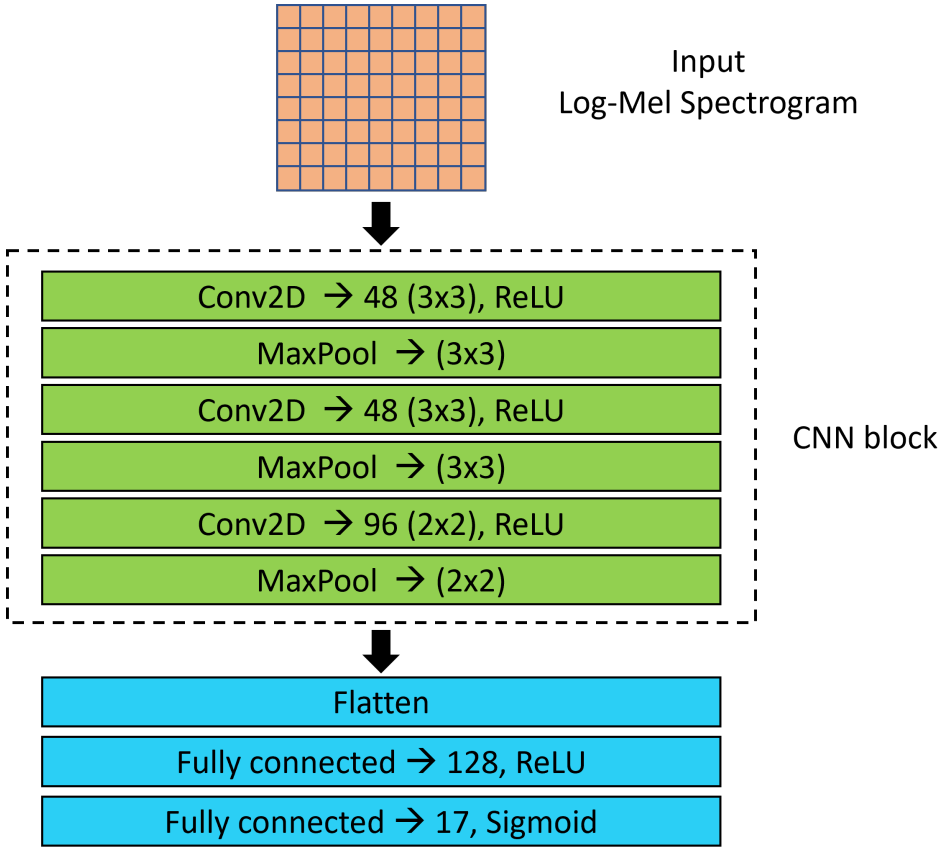


Figure 3.8 – The neural network architecture consists of a CNN block followed by a fully connected layer.

3.4.3 RNN model

The second deep learning model consists of an RNN model. It includes two layers of LSTM, chosen for their ability to control what information to keep and what to forget, thereby preserving important information. The first LSTM layer has 128 units and is bidirectional. This layer returns the full sequence of outputs. The second layer also has 128 units. Both layers use the Tanh activation function. The data is then fed into a fully connected layer, followed by a dropout layer (0.3) before the output, as depicted in Figure 3.9. A Relu activation function is used for the first dense layer, and a Sigmoid for the final output layer. Additionally, L2 regularization is applied to the fully connected layer.

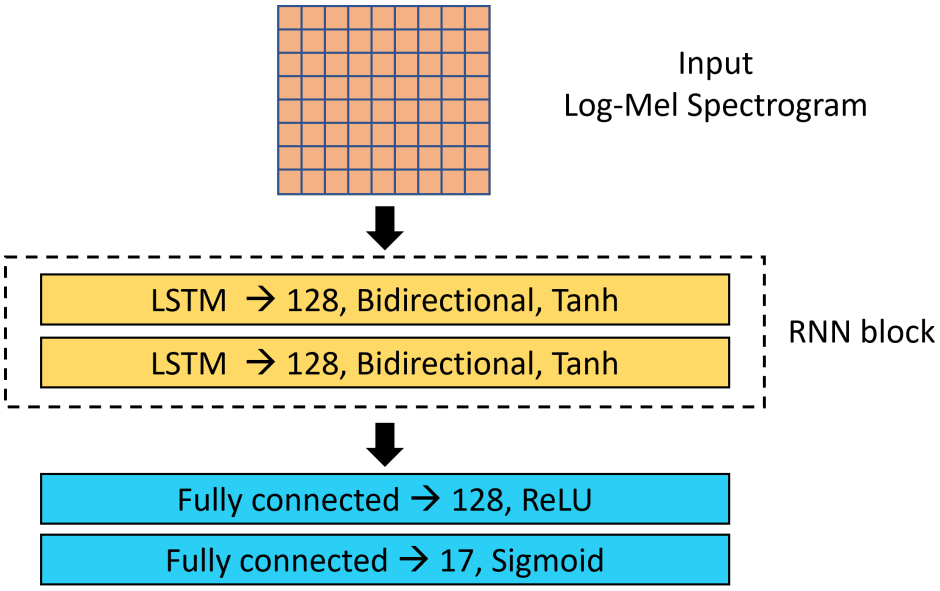


Figure 3.9 – The neural network architecture consists of a RNN block followed by a fully connected layer.

3.4.4 CRNN model

The CNN model is further improved by incorporating an RNN block after the CNN part. The RNN component consists of a bidirectional LSTM layer with 64 units. The output from this RNN layer is then fed into the same fully connected output layers as the previous models. The combination of a CNN and an RNN together forms a hybrid neural network architecture known as a Convolutional Recurrent Neural Network (CRNN). The complete architecture is depicted in Figure 3.10.

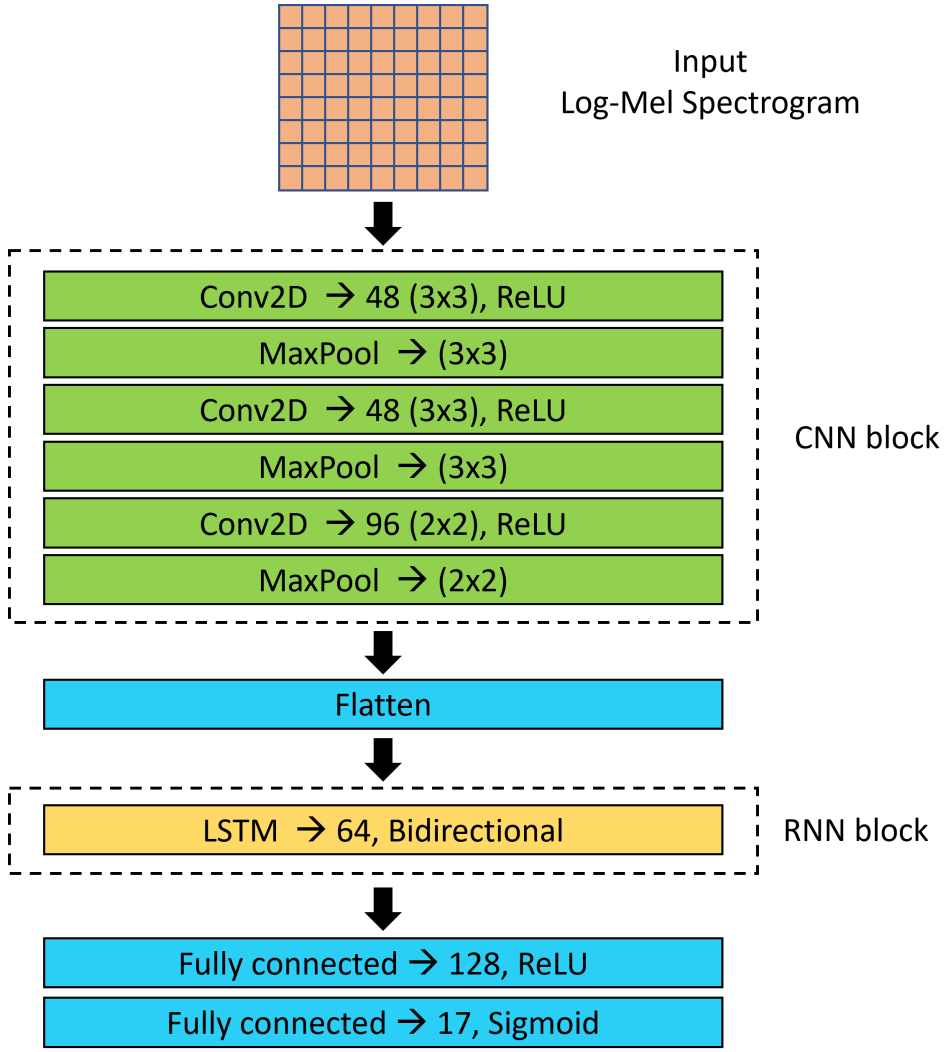


Figure 3.10 – The neural network architecture consists of a CNN layers followed by a LSTM and a fully connected layer.

3.5 Chapter overview

This chapter provided details of the methodology used for constructing the audio dataset of urban sounds in the city of São Paulo. It also discussed the signal pre-processing applied to the data to serve as input for the networks: SVM, CNN, RNN, and CRNN. The next chapter will present the results of this methodology.

4 Results and discussion

This chapter presents the results of the models detailed previously using the configuration of data described in Table 3.2 with the respective discussions. Moreover, some techniques are employed to improve accuracy, including data augmentation and hierarchical classification algorithms.

The same hyperparameters were used for the deep learning models: a learning rate of 0.0001, a batch size of 32, and 120 epochs. An early stop condition controlled the execution, and the lowest validation loss model was recovered in the end. In addition, the Adam optimizer was applied together with the Binary Cross-Entropy loss function.

4.1 Linear SVM results

The linear SVM model experiment was carried out for each configuration from Table 3.2, so it can serve as a reference for the more complex models in the next sections. Table 4.1 presents the accuracies achieved for each configuration of the linear SVM model and the respective standard deviations (σ). Configuration 7 demonstrated the highest performance with an accuracy of 72.0%. However, a two-tailed t-test at a 95% confidence level revealed that this result was not statistically significantly higher than configuration 8 ($t(8) = 2.238, p = 0.056$), which achieved a slightly lower accuracy of 69.8%. The t-test results will be used to indicate whether the differences in accuracy between models are statistically significant. A higher p-value (≥ 0.05) suggests that the difference is likely due to random chance, whereas a lower p-value (< 0.05) indicates a statistically significant difference.

Table 4.1 – Accuracy of the experiments using the linear SVM model.

Configuration	Accuracy (%)	σ (p.p.)
1	58.5	1.23
2	58.9	1.34
3	66.6	0.44
4	66.1	0.40
5	69.1	0.83
6	69.0	0.82
7	72.0	0.95
8	69.8	1.69

4.2 CNN results

The accuracy for each configuration described in Table 3.2 using the CNN model can be seen in Table 4.2 with their respective standard deviations (σ). It is possible to notice no direct relation between the increase of the Mel-spectrogram resolution and higher accuracies. The best result was achieved in Configuration 3 (87.8%) using 32 Mel bands, hop size of 2048, and window length of 4096. However, no statistically significant difference was found between models 3 and 2 ($t(8) = 1.818, p = 0.107$) and models 3 and 4 ($t(8) = 2.156, p = 0.063$).

Table 4.2 – Accuracy of the experiments using the CNN architecture and their respective standard deviations (σ).

Configuration	Accuracy (%)	σ (p.p.)
1	84.2	0.81
2	86.7	0.85
3	87.8	0.87
4	86.4	0.96
5	86.1	1.07
6	82.6	1.68
7	86.0	0.39
8	81.9	0.62

Compared with the SVM, the CNN consistently outperforms across all configurations, exhibiting an improvement of up to 27.8%. This significant performance gain suggests that a non-linear model is necessary for this type of task.

The class-wise precision, recall, and f1-score for each of the eight configurations were calculated and are presented in tables 4.3, 4.4 and 4.5, respectively. The standard deviations are presented alongside. In these tables, the highest value for each class is highlighted. Notably, Configuration 3 emerges with the best overall performance in precision, achieving top results in 6 out of 17 classes. Additionally, Configuration 3 demonstrates superior recall, securing the highest values for six classes, and excels in the f1-score, with seven classes achieving top scores.

Classes like “Insect” and “Alarm” consistently emerged with the best scores across most metrics, reaching precision scores up to 99%. In contrast, low-frequency sound classes such as vehicles tended to exhibit poorer performance, primarily due to the similarities in their spectral profiles, as illustrated in the spectrogram from Section 3.1. Specifically, “Heavy Vehicle (pass-by)” proved particularly challenging to distinguish, achieving only a 68.9% precision score. Meanwhile, other low-frequency sounds located within the first seven positions in the tables generally scored below 90%, except “Heavy Vehicle (idling)”.

Table 4.3 – Precision resulted from the CNN model for each configuration.

	Precision (%)							
Configuration	1	2	3	4	5	6	7	8
Bkg noise	81.9 ±4.5	**82.1** ±2.3	78.5 ±2.1	75.1 ±4.1	71.1 ±5.7	63.8 ±3.2	73.8 ±4.2	65.6 ±2.5
Motorcycle	74.7 ±4.6	**76.5** ±4.3	75.4 ±2.9	70.9 ±6.1	71.8 ±4.9	68.5 ±4.5	68.7 ±7.1	68.8 ±5.2
Car	81.8 ±5.3	**82.5** ±4.0	81.8 ±7.5	78.9 ±7.8	77.1 ±1.9	76.6 ±5.3	69.8 ±5.0	67.5 ±6.5
HV (pass-by)	65.8 ±2.1	66.6 ±3.4	68.6 ±6.9	**68.9** ±3.3	61.0 ±4.8	57.9 ±6.2	67.6 ±6.0	57.4 ±3.4
HV (idling)	95.7 ±2.5	95.1 ±2.2	**95.4** ±1.9	93.4 ±3.1	94.8 ±3.0	89.2 ±3.3	93.5 ±2.2	87.2 ±10.3
Helicopter	80.3 ±9.1	**85.8** ±5.7	**85.8** ±3.2	85.1 ±3.6	80.5 ±2.8	78.4 ±4.3	83.7 ±11.0	79.2 ±6.1
Air conditioner	78.5 ±6.1	84.0 ±4.7	**86.7** ±6.3	85.1 ±3.6	86.5 ±5.8	77.3 ±8.8	80.7 ±2.9	78.9 ±5.5
Const (non-imp)	89.2 ±3.5	92.9 ±2.8	91.3 ±4.1	94.2 ±2.0	**95.2** ±2.9	94.0 ±3.4	93.7 ±2.7	88.4 ±3.1
Const (imp)	88.0 ±4.5	93.4 ±1.2	93.1 ±2.5	**94.7** ±3.9	92.3 ±3.1	91.6 ±2.9	92.8 ±4.6	88.7 ±4.9
Rain	93.3 ±1.7	92.7 ±2.7	**95.5** ±2.6	92.9 ±6.7	91.5 ±5.2	91.8 ±2.5	92.1 ±3.7	89.7 ±2.3
Voice	84.1 ±2.4	86.9 ±2.7	88.6 ±3.3	88.5 ±4.1	**93.6** ±1.8	91.5 ±1.7	91.9 ±2.5	88.1 ±5.8
Music	77.3 ±6.1	79.5 ±5.5	86.5 ±7.7	81.8 ±6.2	85.8 ±2.1	80.8 ±4.0	**89.6** ±1.9	86.1 ±2.9
Dog	86.7 ±2.6	89.6 ±2.4	93.7 ±1.6	92.4 ±2.2	94.0 ±1.3	91.6 ±2.9	**95.0** ±3.0	93.1 ±3.0
Insect	95.7 ±1.6	98.3 ±1.1	**98.6** ±0.7	96.1 ±2.7	96.7 ±3.3	96.3 ±2.7	95.5 ±4.3	96.6 ±2.3
Bird	90.5 ±0.6	92.6 ±3.7	94.1 ±1.8	91.8 ±3.5	94.6 ±2.4	91.6 ±2.4	**96.3** ±3.1	92.3 ±2.4
Alarm	93.7 ±2.1	96.6 ±3.0	97.8 ±1.3	**99.0** ±0.8	97.2 ±2.2	96.8 ±1.7	97.8 ±0.8	96.0 ±2.7
Siren	85.0 ±3.5	88.9 ±5.1	**93.1** ±3.2	91.2 ±3.9	92.8 ±1.8	85.9 ±4.0	90.9 ±2.8	86.7 ±2.2
Macro avg.	84.8 ±0.9	87.3 ±0.9	**88.5** ±0.8	87.1 ±0.8	86.9 ±0.8	83.7 ±1.5	86.7 ±0.6	83.0 ±0.5
Weighted avg.	84.6 ±0.9	87.0 ±1.0	**88.2** ±0.8	86.8 ±0.9	86.7 ±0.9	83.4 ±1.5	86.5 ±0.5	82.6 ±0.5

Table 4.4 – Recall resulted by the CNN model for each configuration.

	Recall (%)							
Configuration	1	2	3	4	5	6	7	8
Bkg noise	88.6 ±3.4	**90.6** ±3.7	89.3 ±4.7	87.7 ±2.6	86.4 ±2.8	86.0 ±3.6	85.4 ±5.3	87.9 ±1.3
Motorcycle	68.0 ±7.9	72.9 ±5.9	**74.8** ±4.5	70.6 ±5.5	66.7 ±5.2	62.7 ±7.6	69.9 ±3.7	69.3 ±4.5
Car	**77.9** ±5.0	**77.9** ±7.3	77.6 ±2.9	77.3 ±4.4	76.1 ±4.5	74.9 ±4.2	77.0 ±3.3	70.8 ±6.3
HV (pass-by)	66.8 ±8.3	**70.1** ±8.1	**70.1** ±4.5	63.8 ±6.7	64.1 ±2.8	59.7 ±5.9	54.2 ±6.9	50.4 ±2.7
HV (idling)	97.1 ±1.2	97.8 ±1.3	**98.1** ±1.9	96.5 ±1.9	96.5 ±1.9	94.0 ±3.5	94.0 ±3.2	91.4 ±1.9
Helicopter	79.3 ±4.0	78.7 ±1.2	78.7 ±3.6	**83.4** ±0.8	79.6 ±8.3	75.9 ±4.0	82.1 ±5.4	72.4 ±3.6
Air conditioner	76.7 ±5.3	81.4 ±1.7	**84.4** ±5.2	80.0 ±3.7	75.6 ±6.1	65.0 ±6.7	79.7 ±6.2	68.9 ±4.7
Const (non-imp)	84.9 ±3.9	89.2 ±2.1	**92.4** ±1.4	91.5 ±4.2	90.0 ±2.4	88.5 ±2.3	89.5 ±3.7	83.9 ±4.4
Const (imp)	93.7 ±3.8	**98.0** ±1.9	97.0 ±3.0	96.0 ±4.1	97.7 ±0.8	95.7 ±2.0	97.4 ±3.8	94.7 ±6.7
Rain	90.2 ±2.3	93.8 ±4.3	94.1 ±3.8	92.5 ±3.6	**95.4** ±3.3	87.6 ±3.9	92.8 ±2.9	90.5 ±3.9
Voice	88.9 ±4.3	89.7 ±3.5	91.5 ±1.5	87.7 ±4.0	91.4 ±4.3	85.2 ±4.9	**93.5** ±2.5	90.2 ±2.0
Music	75.8 ±5.2	82.3 ±7.3	85.6 ±4.0	85.6 ±2.5	88.3 ±2.4	82.3 ±5.3	**88.6** ±1.9	81.0 ±5.5
Dog	82.5 ±3.6	85.0 ±4.1	86.1 ±3.5	86.7 ±3.4	87.8 ±3.2	83.3 ±1.2	**88.6** ±1.0	84.2 ±3.7
Insect	94.1 ±2.8	**98.0** ±1.9	96.0 ±2.7	96.0 ±2.7	94.7 ±2.2	93.1 ±2.2	94.7 ±1.6	90.8 ±3.4
Bird	89.3 ±4.8	92.1 ±2.8	**93.7** ±3.9	92.5 ±3.0	92.1 ±4.1	92.1 ±3.7	89.3 ±4.7	90.2 ±4.6
Alarm	92.5 ±2.7	93.8 ±2.4	96.6 ±2.3	96.2 ±1.9	95.3 ±2.2	93.1 ±1.6	**96.9** ±2.6	95.3 ±1.4
Siren	88.3 ±3.8	86.1 ±6.1	89.2 ±2.0	88.6 ±4.1	88.3 ±3.2	87.5 ±4.4	**90.8** ±3.0	83.6 ±8.5
Macro avg.	84.4 ±0.8	86.9 ±0.8	**88.0** ±0.9	86.6 ±0.9	86.2 ±1.2	82.7 ±1.8	86.1 ±0.4	82.1 ±0.6
Weighted avg.	84.2 ±0.8	86.7 ±0.9	**87.8** ±0.9	86.4 ±1.0	86.1 ±1.1	82.6 ±1.7	86.0 ±0.4	81.9 ±0.6

The normalized confusion matrix of the best configuration (Configuration 3) is depicted in Figure 4.1, highlighting prominent class confusions. As expected, "Heavy vehicle (pass-by)" is frequently misclassified as "Background noise" (7%), "Motorcycle" (7%), and "Car" (8%). Significant errors also occur when 14% of cars and 12% of motorcycles are incorrectly identified as heavy vehicles. This underscores the challenge in distinguishing low-frequency sounds, which often exhibit spectral similarities. Additionally, among the other classes, it is important to point out that 5% of "Voice" samples are mistaken for "Music", and conversely, 7% of music samples are misidentified as voice, which is natural by the presence of voice in music.

Table 4.5 – F1-score resulted by the CNN model for each configuration.

	F1-score (%)							
Configuration	1	2	3	4	5	6	7	8
Bkg noise	85.0 ±3.5	**86.1** ±2.4	83.4 ±1.0	80.8 ±2.1	77.8 ±3.3	73.2 ±3.0	79.0 ±1.8	75.1 ±1.4
Motorcycle	70.7 ±2.7	74.4 ±3.2	**75.0** ±3.0	70.4 ±3.2	69.0 ±4.0	65.0 ±3.7	68.9 ±2.0	68.8 ±1.8
Car	79.6 ±3.7	**79.7** ±2.6	79.5 ±4.6	77.9 ±5.0	76.5 ±1.8	75.6 ±3.9	73.1 ±2.7	68.6 ±2.9
HV (pass-by)	66.0 ±3.7	67.9 ±2.5	**69.1** ±3.9	66.0 ±3.8	62.3 ±1.5	58.5 ±4.1	59.7 ±3.3	53.6 ±2.0
HV (idling)	96.4 ±1.2	96.4 ±1.2	**96.7** ±1.3	94.9 ±1.5	95.6 ±1.0	91.5 ±3.0	93.7 ±1.0	88.8 ±4.7
Helicopter	79.3 ±3.3	82.0 ±2.3	82.0 ±2.2	**84.2** ±2.1	79.8 ±4.4	77.1 ±3.9	82.2 ±4.5	75.4 ±3.1
Air conditioner	77.3 ±3.7	82.6 ±1.8	**85.4** ±4.8	82.4 ±2.1	80.4 ±4.1	70.0 ±2.9	80.1 ±3.2	73.3 ±1.7
Const (non-imp)	86.8 ±2.0	91.0 ±1.9	91.8 ±1.8	**92.8** ±2.5	92.5 ±1.8	91.1 ±1.5	91.5 ±2.2	85.9 ±1.8
Const (imp)	90.6 ±1.6	**95.6** ±1.4	95.0 ±2.0	95.2 ±2.0	94.9 ±1.8	93.6 ±1.3	94.9 ±2.2	91.3 ±3.3
Rain	91.7 ±1.2	93.2 ±2.7	**94.7** ±1.7	92.5 ±3.1	93.4 ±3.7	89.6 ±2.8	92.4 ±2.3	90.1 ±2.9
Voice	86.4 ±2.4	88.2 ±2.2	90.0 ±2.4	88.0 ±3.3	92.5 ±2.0	88.1 ±2.3	**92.6** ±1.5	89.1 ±3.3
Music	76.2 ±1.7	80.6 ±4.4	85.7 ±3.2	83.5 ±3.7	87.0 ±1.4	81.5 ±4.1	**89.1** ±1.5	83.3 ±2.5
Dog	84.5 ±2.4	87.1 ±2.1	89.7 ±1.8	89.4 ±2.1	90.8 ±2.1	87.2 ±1.2	**91.7** ±1.6	88.3 ±1.8
Insect	94.8 ±1.5	**98.2** ±1.4	97.3 ±1.5	96.0 ±2.4	95.7 ±2.5	94.6 ±2.0	95.1 ±2.0	93.5 ±1.3
Bird	89.8 ±2.2	92.3 ±1.7	**93.8** ±1.7	92.0 ±0.8	93.3 ±1.8	91.8 ±2.0	92.6 ±3.3	91.2 ±2.9
Alarm	93.1 ±1.9	95.1 ±1.5	97.2 ±1.8	**97.6** ±0.9	96.2 ±0.6	94.9 ±1.2	97.3 ±1.5	95.6 ±1.8
Siren	86.5 ±2.3	87.4 ±4.9	**91.1** ±2.3	89.7 ±1.8	90.4 ±1.5	86.5 ±2.1	90.8 ±2.4	84.8 ±3.8
Macro avg.	84.4 ±0.8	86.9 ±0.8	**88.1** ±0.8	86.7 ±0.9	86.4 ±1.0	82.9 ±1.7	86.1 ±0.5	82.2 ±0.5
Weighted avg.	84.2 ±0.8	86.7 ±0.8	**87.9** ±0.8	86.4 ±1.0	86.2 ±1.0	82.7 ±1.6	86.0 ±0.4	81.9 ±0.6

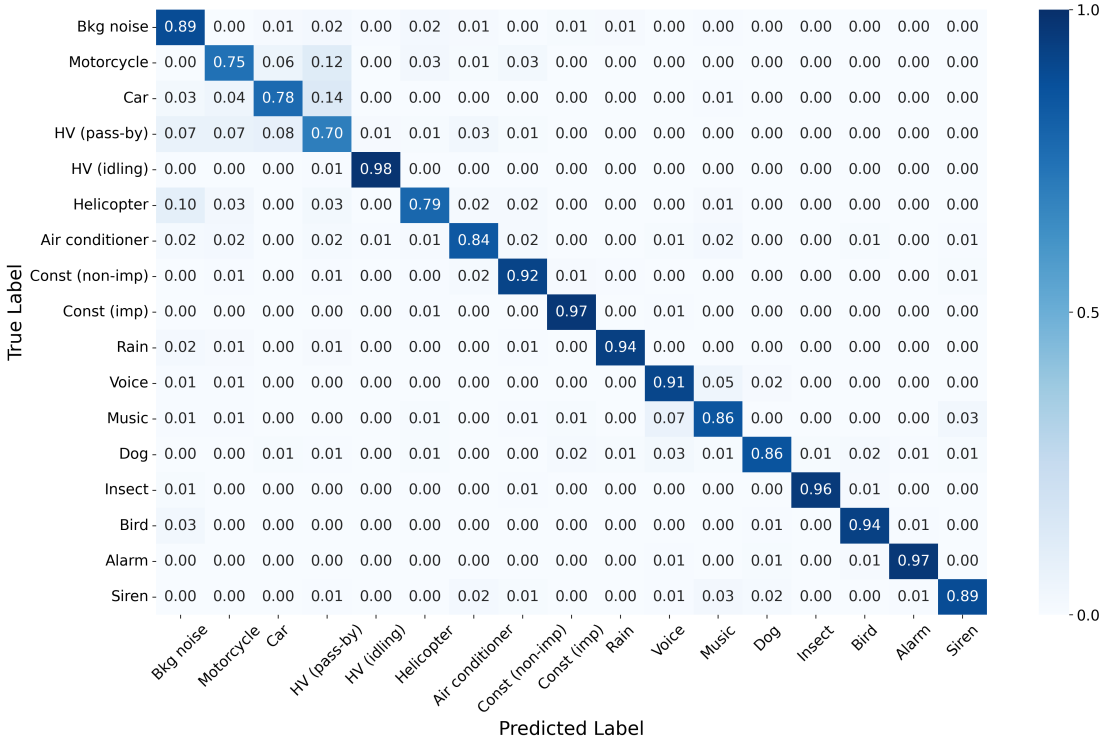


Figure 4.1 – Normalized confusion matrix resulted from using the CNN model and the configuration 3 (32 Mel bands, hop size of 2048 and window length of 4096).

4.3 RNN results

In Table 4.6, the accuracy for the RNN model in each configuration described in Table 3.2 and their respective standard deviations (σ) are shown. As seen in the CNN model, the accuracy does not increase with Mel spectrogram resolution and the best performance was yielded by Configuration 7 (85.3%). In a direct comparison with the CNN

model, configurations 2, 3, 4, and 5 showed a significant decrease in accuracy. However, the two-tailed t-tests for configurations 1, 6, 7, and 8 yielded the following results, respectively: $t(8) = -7.813, p < 0.05$; $t(8) = -7.712, p < 0.05$; $t(8) = -4.074, p < 0.05$; and $t(8) = -3.448, p < 0.05$. Therefore, configurations 1, 6, 7, and 8 are equivalent in both CNN and RNN models.

In tables 4.7, 4.8 and 4.9, the class-wise precision, recall, and f1-score for every tested configuration were calculated and are presented, respectively, with the highest value highlighted for each class. Configuration 7 stands out among all the configurations in the class-wise comparison, exhibiting the best overall performance in precision by achieving the highest results in 9 out of 17 classes. Additionally, Configuration 7 demonstrates superior recall, attaining the top values for seven classes, and excels in the f1-score, with eight classes achieving the best scores.

Table 4.6 – Accuracy of the experiments using the RNN architecture and their respective standard deviations (σ).

Configuration	Accuracy (%)	σ (p.p.)
1	83.2	1.16
2	82.2	0.80
3	83.5	0.73
4	83.3	1.18
5	84.0	0.58
6	82.1	1.35
7	85.3	1.08
8	82.0	1.36

Table 4.7 – Precision resulted by the RNN model for each configuration.

	Precision (%)							
Configuration	1	2	3	4	5	6	7	8
Bkg noise	**83.9** ±3.5	81.9 ±2.4	78.7 ±2.9	76.8 ±3.6	75.9 ±3.7	72.3 ±3.3	76.6 ±5.3	69.5 ±1.6
Motorcycle	64.9 ±7.4	64.2 ±7.5	65.1 ±4.8	66.4 ±5.3	71.5 ±6.7	66.6 ±3.4	**77.0** ±5.2	66.9 ±3.3
Car	77.6 ±7.5	77.7 ±4.4	76.8 ±5.5	81.9 ±3.4	**82.6** ±4.3	72.0 ±8.0	80.2 ±3.8	72.3 ±5.2
HV (pass-by)	**63.5** ±6.9	58.2 ±9.3	61.4 ±4.0	62.0 ±4.4	59.6 ±4.9	63.3 ±6.0	58.4 ±3.8	57.6 ±1.7
HV (idling)	94.0 ±3.4	94.5 ±3.9	94.1 ±3.6	93.2 ±2.9	94.4 ±3.1	89.4 ±3.7	**95.2** ±3.9	93.4 ±1.7
Helicopter	74.3 ±4.6	72.7 ±2.3	74.1 ±8.7	76.0 ±2.8	77.3 ±6.1	77.1 ±6.5	**85.7** ±4.5	80.8 ±6.3
Air conditioner	81.9 ±7.6	79.5 ±4.6	81.6 ±5.8	79.4 ±8.3	84.3 ±2.5	82.8 ±4.4	**86.5** ±4.3	83.0 ±5.9
Const (non-imp)	91.2 ±6.5	88.1 ±2.6	**93.8** ±3.9	94.3 ±3.8	91.9 ±5.9	90.0 ±4.1	93.3 ±4.8	91.4 ±3.2
Const (imp)	91.1 ±2.8	88.1 ±4.9	87.7 ±2.3	89.8 ±4.7	90.9 ±4.8	**91.2** ±2.7	90.8 ±3.4	89.4 ±7.2
Rain	91.5 ±1.8	90.0 ±3.3	**94.3** ±3.5	90.2 ±3.2	89.8 ±3.8	88.5 ±1.9	89.2 ±1.8	91.2 ±2.7
Voice	83.2 ±5.9	82.3 ±7.7	82.1 ±3.3	84.0 ±1.1	83.7 ±3.9	78.8 ±2.4	**88.3** ±2.3	81.3 ±4.3
Music	73.7 ±6.0	78.7 ±2.2	78.6 ±4.4	78.0 ±6.1	81.9 ±4.2	83.4 ±5.2	**85.2** ±4.1	77.9 ±4.7
Dog	84.4 ±2.5	84.5 ±4.8	86.0 ±5.0	86.0 ±4.4	**88.9** ±1.9	84.3 ±4.6	84.5 ±4.9	87.9 ±6.1
Insect	96.4 ±1.8	97.6 ±2.2	97.3 ±1.7	95.4 ±3.7	96.3 ±2.8	95.6 ±0.8	**98.9** ±1.4	96.2 ±2.8
Bird	91.6 ±4.2	91.5 ±3.0	92.4 ±3.4	91.5 ±5.0	90.1 ±1.4	90.1 ±3.9	**93.3** ±4.2	90.6 ±5.9
Alarm	95.0 ±3.7	94.1 ±1.7	97.2 ±2.0	95.7 ±3.4	96.6 ±2.6	97.1 ±1.9	**98.1** ±1.8	95.2 ±2.1
Siren	87.0 ±4.5	86.6 ±4.9	**90.9** ±3.2	86.5 ±4.1	89.0 ±4.6	86.6 ±3.3	86.3 ±4.7	87.5 ±1.3
Macro avg.	83.8 ±1.2	82.9 ±0.7	84.3 ±0.6	83.9 ±1.1	85.0 ±0.4	82.9 ±1.3	**86.3** ±1.1	83.1 ±1.1
Weighted avg.	83.7 ±1.2	82.7 ±0.7	84.0 ±0.6	83.7 ±1.2	84.7 ±0.5	82.6 ±1.4	**86.0** ±1.1	82.7 ±1.2

Table 4.8 – Recall resulted by the RNN model for each configuration.

Recall (%)								
Configuration	1	2	3	4	5	6	7	8
Bkg noise	85.7 ±3.6	86.2 ±2.7	85.2 ±2.1	83.6 ±4.9	83.3 ±4.8	86.4 ±2.5	86.7 ±3.1	**88.6** ±4.1
Motorcycle	65.0 ±11.8	57.8 ±7.6	65.1 ±8.3	**68.6** ±7.2	64.4 ±3.1	65.7 ±11.8	64.1 ±6.9	58.2 ±8.3
Car	74.6 ±5.3	72.3 ±2.4	**75.2** ±2.3	71.7 ±2.8	69.0 ±5.1	**75.2** ±3.7	74.3 ±5.9	72.0 ±3.9
HV (pass-by)	66.6 ±3.8	67.4 ±5.8	64.7 ±6.8	63.0 ±5.0	71.2 ±6.1	58.6 ±4.3	**72.9** ±5.3	62.2 ±3.4
HV (idling)	**97.1** ±1.9	95.2 ±1.4	94.6 ±2.2	94.0 ±3.2	93.7 ±2.2	94.9 ±2.1	96.5 ±2.7	89.8 ±2.9
Helicopter	74.9 ±7.3	75.3 ±5.7	79.6 ±5.2	75.9 ±5.1	82.2 ±6.3	79.6 ±4.0	**82.5** ±7.7	79.6 ±2.1
Air conditioner	80.3 ±1.4	78.3 ±1.1	79.7 ±4.1	83.3 ±4.7	**85.8** ±1.8	78.9 ±4.9	82.2 ±3.2	81.7 ±4.2
Const (non-imp)	85.8 ±5.7	86.8 ±1.5	85.3 ±2.6	87.5 ±3.0	88.3 ±2.4	85.8 ±1.0	**89.0** ±1.5	84.9 ±3.5
Const (imp)	90.8 ±2.9	89.1 ±4.2	93.7 ±3.5	94.4 ±4.9	91.7 ±2.1	91.1 ±3.8	**95.4** ±2.9	92.1 ±3.2
Rain	90.6 ±3.6	89.3 ±3.3	90.5 ±4.9	**91.9** ±3.6	89.2 ±7.4	87.6 ±5.5	90.9 ±4.8	89.2 ±6.1
Voice	84.7 ±5.2	84.7 ±6.2	86.2 ±6.4	85.7 ±6.4	86.4 ±5.4	83.6 ±4.8	**86.9** ±4.3	85.2 ±2.9
Music	75.3 ±3.1	75.0 ±3.5	77.7 ±4.5	74.2 ±6.3	80.2 ±2.5	75.8 ±6.1	**82.6** ±4.4	78.0 ±2.9
Dog	82.5 ±3.0	78.3 ±6.8	80.0 ±2.6	79.7 ±4.0	81.7 ±1.4	79.7 ±4.5	**86.9** ±3.7	81.1 ±3.4
Insect	**96.0** ±3.3	93.7 ±1.9	94.1 ±1.7	93.4 ±2.1	93.1 ±1.9	92.1 ±2.8	90.4 ±4.3	90.4 ±4.3
Bird	90.6 ±2.8	89.3 ±4.9	89.6 ±4.0	**91.5** ±1.9	86.2 ±4.7	85.2 ±6.1	88.1 ±6.5	84.3 ±6.6
Alarm	92.8 ±1.9	94.7 ±1.6	**95.6** ±2.7	95.3 ±1.7	94.4 ±0.8	92.2 ±0.0	94.7 ±2.5	92.8 ±1.2
Siren	84.2 ±1.9	85.0 ±5.5	85.6 ±6.0	87.2 ±3.4	**88.6** ±3.8	86.1 ±4.2	86.7 ±4.3	85.3 ±4.4
Macro avg.	83.4 ±1.2	82.3 ±0.9	83.7 ±0.7	83.6 ±1.1	84.1 ±0.6	82.3 ±1.5	**85.3** ±1.0	82.1 ±1.4
Weighted avg.	83.2 ±1.2	82.2 ±0.8	83.5 ±0.7	83.3 ±1.2	84.0 ±0.6	82.1 ±1.4	**85.3** ±1.1	82.0 ±1.4

Table 4.9 – F1-score resulted by the RNN model for each configuration.

F1-score (%)								
Configuration	1	2	3	4	5	6	7	8
Bkg noise	**84.7** ±1.9	84.0 ±2.2	81.8 ±2.1	79.9 ±3.5	79.2 ±2.1	78.7 ±1.6	81.3 ±3.6	77.9 ±1.5
Motorcycle	64.5 ±8.6	60.4 ±5.4	64.6 ±4.2	67.2 ±4.7	67.7 ±4.7	65.7 ±7.3	**69.6** ±3.8	61.9 ±4.5
Car	75.8 ±4.5	74.8 ±3.2	75.8 ±2.1	76.4 ±2.8	74.9 ±1.6	73.2 ±4.0	**76.9** ±2.3	72.0 ±3.1
HV (pass-by)	**64.8** ±4.3	62.1 ±6.7	62.8 ±3.4	62.2 ±2.4	64.5 ±1.7	60.6 ±3.2	64.7 ±3.6	59.8 ±2.2
HV (idling)	95.5 ±1.0	94.8 ±2.7	94.3 ±2.3	93.5 ±1.8	94.0 ±1.9	92.0 ±1.9	**95.8** ±1.7	91.6 ±1.5
Helicopter	74.4 ±4.4	73.9 ±3.4	76.3 ±3.9	75.9 ±3.9	79.2 ±2.2	78.1 ±3.5	**83.8** ±4.1	80.1 ±3.6
Air conditioner	80.9 ±3.3	78.8 ±2.2	80.5 ±2.9	80.9 ±3.8	**85.0** ±1.3	80.5 ±1.2	84.2 ±3.0	82.2 ±4.0
Const (non-imp)	88.1 ±3.3	87.4 ±1.8	89.3 ±1.8	90.8 ±2.9	90.0 ±3.6	87.8 ±2.0	**91.1** ±3.0	88.0 ±2.5
Const (imp)	90.9 ±2.1	88.4 ±1.7	90.6 ±2.5	91.8 ±2.1	91.2 ±1.8	91.1 ±2.4	92.9 ±2.0	90.6 ±4.8
Rain	91.0 ±1.3	89.5 ±2.2	**92.3** ±3.5	91.0 ±2.6	89.3 ±3.7	88.0 ±2.8	89.9 ±2.3	90.0 ±3.0
Voice	83.6 ±2.3	82.9 ±1.3	83.9 ±3.0	84.7 ±2.9	84.8 ±1.1	81.1 ±2.3	**87.5** ±2.5	83.1 ±2.9
Music	74.2 ±2.5	76.7 ±2.0	78.1 ±4.1	75.8 ±4.5	81.0 ±2.2	79.3 ±5.2	**83.8** ±4.0	77.9 ±3.5
Dog	83.4 ±2.7	81.1 ±4.2	82.8 ±2.6	82.7 ±3.9	85.1 ±1.2	81.7 ±1.4	**85.6** ±2.2	84.2 ±2.9
Insect	**96.2** ±1.7	95.6 ±1.6	95.6 ±1.3	94.4 ±2.3	94.6 ±2.0	93.8 ±1.5	94.4 ±2.6	93.2 ±3.0
Bird	91.0 ±3.0	90.3 ±2.4	90.9 ±2.2	**91.4** ±2.1	88.1 ±2.7	87.4 ±2.6	90.4 ±3.5	87.1 ±4.5
Alarm	93.9 ±2.6	94.4 ±1.3	**96.4** ±1.3	95.5 ±1.7	95.4 ±1.0	94.6 ±0.9	96.3 ±1.3	94.0 ±1.2
Siren	85.5 ±2.7	85.6 ±3.1	88.0 ±3.8	86.8 ±2.5	**88.8** ±3.4	86.3 ±3.3	86.3 ±2.0	86.3 ±2.6
Macro avg.	83.4 ±1.2	82.4 ±0.8	83.8 ±0.7	83.6 ±1.1	84.3 ±0.5	82.3 ±1.4	**85.6** ±1.0	82.3 ±1.3
Weighted avg.	83.2 ±1.2	82.2 ±0.7	83.5 ±0.7	83.4 ±1.2	84.1 ±0.5	82.1 ±1.3	**85.4** ±1.1	82.1 ±1.3

As seen in the CNN architecture, classes “Insect” and “Alarm” achieved the top scores, reaching precision scores up to 98.9% in the precision. Low-frequency sound classes tended to exhibit poorer performance, particularly the “Heavy Vehicle (pass-by)” class, which achieved the best precision score of only 63.5%, obtained by Configuration 1. It is important to highlight that there is a significant disparity between the precision and recall scores for all configurations in this class. For instance, Configuration 7 scored only 58.4% in precision and 72.9% in recall, indicating that many samples were incorrectly classified as “Heavy Vehicle (pass-by).”

4.4 CRNN results

Using The CRNN model, all configurations have shown improvement compared to previous models, as depicted in Table 4.10. Configuration 8 presented the highest leap: 9.6% of enhancement. This same model also reached the best overall accuracy among all: 91.5%. Unlike the CNN model, there is a noticeable gradual improvement in accuracy while the Mel spectrogram resolution increases. However, it is important to notice that the increment between successive configurations is small, Model 8 has not shown statistically significant difference compared to Model 7 ($t(8) = 0.273, p = 0.792$), Model 6 ($t(8) = 1.877, p = 0.097$), Model 5 ($t(8) = 0.922, p = 0.384$) and Model 3 ($t(8) = 2.100, p = 0.069$). The class-wise precision, recall and f1-score are depicted in tables 4.3, 4.4 and 4.5, respectively, with the best results of each class being highlighted. Configuration 8 has graded top score in 7 out of 17 classes in each of the metrics.

"Alarm" has achieved the best class result with over 99% in the metrics, while "Heavy vehicle (pass-by)" presented the worse results again, with top scores ranging from 75% to 80%. However, this still represents crucial improvement compared to the CNN model which best result was only 70.1% on recall.

Table 4.10 – Accuracy of the experiments using the CRNN architecture and their respective standard deviations (σ).

Configuration	Accuracy (%)	σ (p.p.)
1	85.6	1.20
2	88.4	1.04
3	89.7	1.41
4	89.2	1.20
5	90.7	1.49
6	90.1	1.09
7	91.3	1.15
8	91.5	0.95

The normalized confusion matrix of the best configuration (Configuration 8) is depicted in Figure 4.2, highlighting prominent class confusion. As expected, although the confusion still exists in the low-frequency sounds, it is in a smaller degree. Compared to the CNN model, the "Heavy vehicle (pass-by)" incorrect assignment as "Background noise" dropped from 7% to 2%, "Motorcycle" from 7% to 6%, and "Car" remained at 8%. However, a significant error of 15% still occurs when cars are incorrectly identified as heavy vehicles, which raised 1% compared to the previous model. Another improvement can be seen when 5% of voice samples mistaken for music dropped to 1%, and conversely, 7% of music samples misidentified as voice lowered to 2%.

Poorer performance in heavy vehicles has already been found in sound classification experiments. Ashhad *et al.* (2023) conducted a study using the IDMT traffic

Table 4.11 – Precision resulted by the CRNN model for each configuration.

	Precision (%)							
Configuration	1	2	3	4	5	6	7	8
Bkg noise	83.5 ±8.4	80.6 ±3.3	84.6 ±3.3	82.6 ±2.0	83.7 ±3.1	81.6 ±4.7	83.7 ±2.6	**85.0** ±2.2
Motorcycle	75.7 ±8.0	79.5 ±4.1	77.2 ±5.9	80.8 ±6.7	76.9 ±7.3	**83.7** ±7.5	79.5 ±2.3	82.8 ±5.4
Car	84.9 ±2.0	**85.2** ±5.0	84.0 ±6.0	81.2 ±2.7	81.8 ±4.0	80.5 ±2.4	78.1 ±4.4	84.0 ±4.8
HV (pass-by)	71.4 ±2.4	73.0 ±8.5	**76.9** ±4.6	72.5 ±5.1	76.1 ±6.4	73.7 ±3.3	75.5 ±5.8	72.6 ±4.3
HV (idling)	96.3 ±1.5	97.3 ±2.9	94.8 ±2.0	95.1 ±2.8	95.7 ±1.9	96.0 ±2.4	**97.5** ±2.1	97.2 ±2.6
Helicopter	74.5 ±2.5	86.9 ±6.1	87.2 ±6.1	89.1 ±5.0	88.4 ±4.6	89.3 ±5.3	**90.5** ±3.9	87.7 ±6.8
Air conditioner	86.3 ±5.6	85.4 ±3.9	89.1 ±4.3	83.7 ±2.7	87.4 ±3.4	87.4 ±5.1	90.9 ±1.9	**91.1** ±1.4
Const (non-imp)	88.9 ±2.4	90.9 ±3.9	94.6 ±2.0	95.1 ±3.7	**97.5** ±1.6	95.9 ±2.5	95.4 ±2.6	97.0 ±2.2
Const (imp)	92.0 ±2.0	94.7 ±3.2	93.9 ±3.7	95.6 ±2.0	95.6 ±2.5	95.4 ±3.5	96.1 ±1.6	**97.1** ±2.3
Rain	93.7 ±1.0	94.3 ±3.2	94.5 ±2.9	97.7 ±2.0	96.2 ±3.4	96.4 ±1.2	**98.0** ±0.6	97.8 ±1.9
Voice	85.8 ±4.1	89.6 ±2.9	90.1 ±3.8	87.2 ±3.7	94.0 ±3.4	93.3 ±3.1	**95.8** ±3.5	94.9 ±2.2
Music	75.0 ±5.4	84.2 ±5.5	88.8 ±5.7	89.5 ±2.9	92.3 ±1.8	87.8 ±3.5	93.7 ±2.1	**93.9** ±2.8
Dog	89.1 ±5.0	91.5 ±4.8	93.0 ±1.6	90.9 ±1.8	93.8 ±3.4	94.3 ±3.4	93.9 ±0.9	**94.8** ±1.1
Insect	96.4 ±1.8	97.7 ±1.9	98.0 ±1.9	**98.7** ±1.9	97.7 ±1.3	98.3 ±1.8	98.3 ±1.1	97.3 ±2.7
Bird	92.7 ±2.4	95.7 ±4.2	95.7 ±3.1	95.9 ±1.8	96.5 ±2.4	94.7 ±2.1	96.1 ±2.6	**97.8** ±2.4
Alarm	95.2 ±2.2	96.8 ±1.0	96.6 ±1.5	97.8 ±1.6	**99.4** ±0.8	99.1 ±1.2	98.8 ±0.6	**99.4** ±0.8
Siren	85.8 ±4.8	91.9 ±4.7	91.1 ±3.1	91.5 ±2.9	93.4 ±3.3	93.7 ±4.1	**95.7** ±1.7	92.8 ±2.3
Macro avg.	86.3 ±1.1	89.1 ±0.8	90.0 ±1.3	89.7 ±1.1	91.0 ±1.4	90.6 ±1.0	91.6 ±1.0	**91.9** ±1.0
Weighted avg.	86.1 ±1.1	88.8 ±0.9	89.9 ±1.4	89.4 ±1.1	90.8 ±1.4	90.4 ±1.0	91.5 ±1.0	**91.8** ±0.9

Table 4.12 – Recall resulted by the CRNN model for each configuration.

	Recall (%)							
Configuration	1	2	3	4	5	6	7	8
Bkg noise	88.6 ±4.1	**93.5** ±2.1	89.1 ±2.5	92.7 ±3.7	88.6 ±2.4	89.8 ±3.6	93.2 ±1.8	91.3 ±3.7
Motorcycle	73.9 ±6.6	73.5 ±4.8	78.5 ±7.4	74.5 ±2.3	81.4 ±6.8	80.1 ±4.1	82.0 ±3.1	**84.0** ±2.8
Car	77.9 ±4.1	79.0 ±7.1	81.7 ±4.4	81.1 ±1.8	79.1 ±3.4	81.4 ±5.2	**82.6** ±5.7	78.2 ±3.4
HV (pass-by)	71.0 ±5.7	75.3 ±8.4	74.8 ±7.9	72.9 ±5.8	72.1 ±1.4	75.3 ±3.6	71.5 ±2.5	**78.9** ±5.4
HV (idling)	97.8 ±0.8	**98.4** ±1.4	97.8 ±1.6	96.8 ±1.7	97.5 ±1.6	**98.4** ±1.0	96.2 ±1.9	97.5 ±2.6
Helicopter	84.7 ±4.1	80.9 ±6.6	84.6 ±1.5	84.0 ±5.0	89.7 ±4.0	84.3 ±5.5	90.0 ±5.2	**90.3** ±2.3
Air conditioner	78.1 ±4.1	83.9 ±5.3	86.7 ±1.9	86.7 ±2.6	87.2 ±2.0	84.7 ±3.8	**88.1** ±2.1	87.8 ±2.2
Const (non-imp)	87.8 ±2.9	91.2 ±1.4	93.6 ±2.9	92.9 ±2.5	**94.4** ±2.4	94.1 ±1.2	**94.4** ±2.7	92.9 ±1.8
Const (imp)	94.4 ±1.7	97.4 ±1.7	97.7 ±3.2	**98.4** ±1.8	98.0 ±2.4	98.0 ±1.2	98.0 ±1.9	98.0 ±1.6
Rain	91.5 ±4.8	93.2 ±5.2	95.1 ±3.7	95.1 ±3.4	94.8 ±4.1	94.1 ±3.0	**96.1** ±2.2	95.8 ±2.4
Voice	87.4 ±4.6	90.4 ±3.7	94.5 ±1.0	94.0 ±3.2	**96.7** ±1.9	93.0 ±4.4	96.2 ±1.6	**96.7** ±1.3
Music	79.9 ±4.0	85.1 ±2.7	85.9 ±4.5	83.7 ±5.8	91.3 ±4.6	91.0 ±3.9	92.7 ±3.3	**94.0** ±2.4
Dog	80.8 ±3.9	85.0 ±2.8	87.8 ±1.8	88.3 ±3.6	**90.3** ±2.9	90.0 ±1.6	89.7 ±1.7	**90.3** ±4.1
Insect	96.0 ±3.7	**97.7** ±2.5	97.0 ±2.6	97.3 ±1.7	97.0 ±1.9	96.0 ±2.5	97.0 ±2.0	96.0 ±1.7
Bird	90.6 ±1.9	95.0 ±3.4	93.7 ±4.3	94.6 ±4.3	94.7 ±3.5	**95.6** ±2.5	93.7 ±4.6	94.3 ±3.8
Alarm	93.1 ±2.1	95.6 ±1.8	97.8 ±1.6	97.2 ±1.2	98.1 ±1.8	98.1 ±1.2	**99.4** ±0.8	**99.4** ±0.8
Siren	86.4 ±3.2	89.7 ±3.1	90.8 ±3.4	87.5 ±3.9	**92.8** ±3.2	90.0 ±3.8	92.5 ±3.1	91.7 ±3.2
Macro avg.	85.9 ±1.2	88.5 ±1.1	89.8 ±1.5	89.3 ±1.2	90.8 ±1.5	90.2 ±1.1	91.4 ±1.2	**91.6** ±0.9
Weighted avg.	85.6 ±1.2	88.4 ±1.0	89.7 ±1.4	89.2 ±1.2	90.7 ±1.5	90.1 ±1.1	91.3 ±1.2	**91.5** ±1.0

dataset of vehicles with four classes: “Car”, “Motorcycle”, “Truck” and “No vehicle”, where a CNN architecture was implemented for classification. The precision of each class was 88%, 98%, 69%, and 100%, respectively. In a further investigation of the truck data, they conducted a subjective test with humans. They analyzed the wrongly classified samples of trucks by both the Neural Network and the human subjects. It was discovered that there was a correlation between the misclassifications: over 90% of the wrongly classified truck samples by the human subjects were also misclassified by the Neural Network.

Table 4.13 – F1-score resulted by the CRNN model for each configuration.

	F1-score (%)							
Configuration	1	2	3	4	5	6	7	8
Bkg noise	85.7 ±4.6	86.5 ±1.6	86.7 ±1.8	87.3 ±2.4	86.0 ±2.2	85.4 ±2.6	**88.2** ±2.2	88.0 ±2.0
Motorcycle	74.4 ±4.4	76.2 ±3.1	77.5 ±4.4	77.4 ±3.9	78.9 ±5.6	81.6 ±4.3	80.7 ±1.9	**83.2** ±1.9
Car	81.2 ±2.5	81.6 ±2.3	**82.7** ±4.2	81.1 ±1.4	80.4 ±3.1	80.9 ±2.9	80.2 ±4.3	80.9 ±3.6
HV (pass-by)	71.1 ±3.5	73.3 ±2.6	**75.6** ±4.8	72.4 ±3.2	73.9 ±2.6	74.4 ±2.5	73.3 ±3.0	75.5 ±3.4
HV (idling)	97.0 ±1.0	**97.8** ±1.0	96.3 ±1.2	95.9 ±1.0	96.5 ±0.3	97.2 ±1.6	96.8 ±1.0	97.3 ±1.6
Helicopter	79.2 ±1.4	83.6 ±5.1	85.8 ±3.4	86.5 ±4.5	89.0 ±3.7	86.6 ±4.5	**90.1** ±3.3	88.8 ±3.1
Air conditioner	81.7 ±1.7	84.4 ±2.0	87.8 ±2.0	85.1 ±2.1	87.2 ±1.9	85.8 ±2.2	**89.4** ±0.6	**89.4** ±1.1
Const (non-imp)	88.3 ±1.8	91.0 ±2.2	94.1 ±1.7	94.0 ±2.6	**95.9** ±2.0	95.0 ±1.1	94.8 ±1.9	94.9 ±1.6
Const (imp)	93.2 ±0.9	96.0 ±1.5	95.7 ±1.7	96.9 ±0.6	96.8 ±1.4	96.6 ±1.3	97.1 ±1.2	**97.5** ±1.5
Rain	92.5 ±2.4	93.6 ±2.4	94.8 ±2.9	96.3 ±2.3	95.4 ±2.3	95.2 ±1.3	**97.0** ±1.1	96.7 ±0.6
Voice	86.6 ±3.4	90.0 ±2.8	92.2 ±1.9	90.4 ±1.7	95.3 ±2.0	93.0 ±2.5	**96.0** ±2.1	95.8 ±1.5
Music	77.1 ±2.5	84.4 ±1.9	87.2 ±4.1	86.3 ±2.5	91.7 ±2.7	89.3 ±2.9	93.1 ±2.0	**93.9** ±1.3
Dog	84.6 ±2.2	88.1 ±3.5	90.3 ±1.3	89.5 ±1.5	91.9 ±2.5	92.1 ±2.1	91.8 ±0.6	**92.4** ±2.3
Insect	96.2 ±1.8	97.7 ±1.3	97.5 ±1.9	**98.0** ±1.5	97.3 ±1.2	97.2 ±1.8	97.7 ±1.3	96.7 ±2.2
Bird	91.6 ±1.6	95.3 ±3.3	94.6 ±2.0	95.2 ±2.4	95.5 ±2.2	95.1 ±2.2	94.9 ±3.4	**96.0** ±2.5
Alarm	94.2 ±1.8	96.2 ±1.2	97.2 ±1.4	97.5 ±1.1	98.7 ±1.1	98.6 ±0.9	99.1 ±0.3	**99.4** ±0.6
Siren	86.1 ±3.7	90.8 ±3.4	91.0 ±3.2	89.5 ±3.4	93.0 ±2.7	91.7 ±1.3	**94.0** ±1.7	92.2 ±1.6
Macro avg.	85.9 ±1.2	88.6 ±1.0	89.8 ±1.4	89.4 ±1.1	90.8 ±1.4	90.3 ±1.1	91.4 ±1.1	**91.7** ±1.0
Weighted avg.	85.7 ±1.2	88.4 ±1.0	89.7 ±1.4	89.2 ±1.2	90.7 ±1.4	90.2 ±1.1	91.3 ±1.1	**91.5** ±1.0

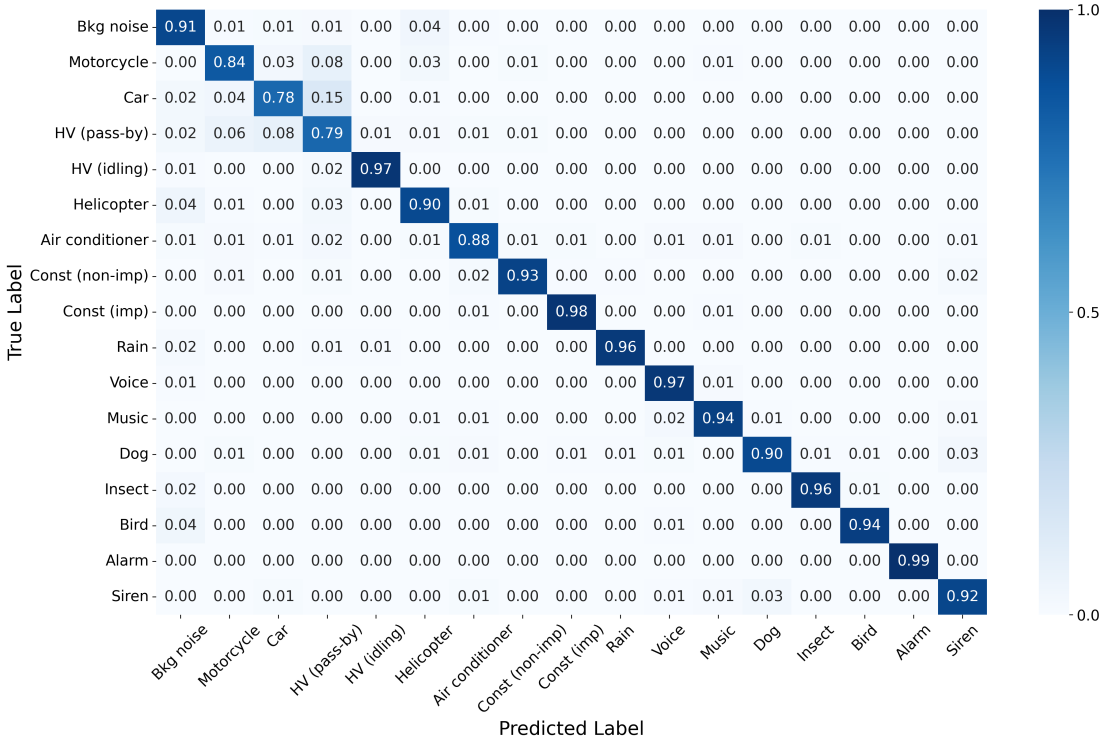


Figure 4.2 – Normalised confusion matrix resulted from using the CRNN model and the configuration 8 (128 Mel bands, hop size of 512, and window length of 1024).

4.5 CRNN with data augmentation results

As an attempt to improve even further the model's classification performance, data augmentation techniques using signal modifications were applied to the dataset with the aid of the Python library Audiomentations (JORDAL, 2022). The applied signal modifications are as follows:

- **Gaussian noise addition:** Adds Gaussian noise to the samples with a random Signal-to-Noise Ratio (SNR) (SNR range: 10 dB to 35 dB) (p=0.8).
- **Time stretch:** Time stretches the signal without changing the pitch (rate range: 0.8 to 1.25) (p=0.5).
- **Pitch shift:** Pitch shifts the sound up or down without changing the tempo (semitone range: -4 to 4) (p=0.5).
- **Sample shift:** Shifts the samples forward or backward with rollover (shift range: -0.5 to 0.5) (p=0.5).

The modifications are applied to the training split. p is the probability of executing the respective augmentation so that a single sample can undergo multiple modifications. A total of 4000 new samples were generated.

Another technique employed was audio mixing (VIDAÑA-VILA *et al.*, 2021), which involves summing audio waveforms or spectrograms to generate new samples. These newly generated samples contain information from both source audios, and their label files were created by aggregating the one-hot encoding values of the original samples. Before mixing, waveforms were normalized to the same energy (RMS = 1) to ensure equal contribution from each source. After mixing, the amplitude was normalized again. This modification was applied to all data classes except for "Background noise", as background noise in urban classification typically needs to be detected without other prominent sounds. Figure 4.3 demonstrates a schematic representation of the audio mixing process, using an alarm (audio 1) and human voice (audio 2) as examples. The figure shows the waveform, Mel spectrogram, and an example of one-hot encoding for each. Over 1000 new samples were generated through this process. Source audios were randomly selected from the entire dataset, allowing for any possible combination of sounds.

Five thousand new files were generated using all data augmentation techniques comprising signal modification and audio mixing. These files were added to the training dataset, in a total of 10864, approximately 1172 for testing and 9692 for training, each run.

As discussed previously, the hybrid model CRNN obtained better results than the CNN or RNN models. Therefore, this was the model chosen for this task using data augmentation. The modifications were explicitly applied to configurations 5, 6, 7, and 8. In Table 4.14, the accuracy with and without data augmentation is exhibited, with the respective standard deviations and the result of the two-tailed t-test comparison.

Three models showed decreased performance, while one demonstrated improvement. However, statistical analysis using a t-test revealed no significant differences in these comparisons. Despite data augmentation techniques often being helpful in increasing reg-

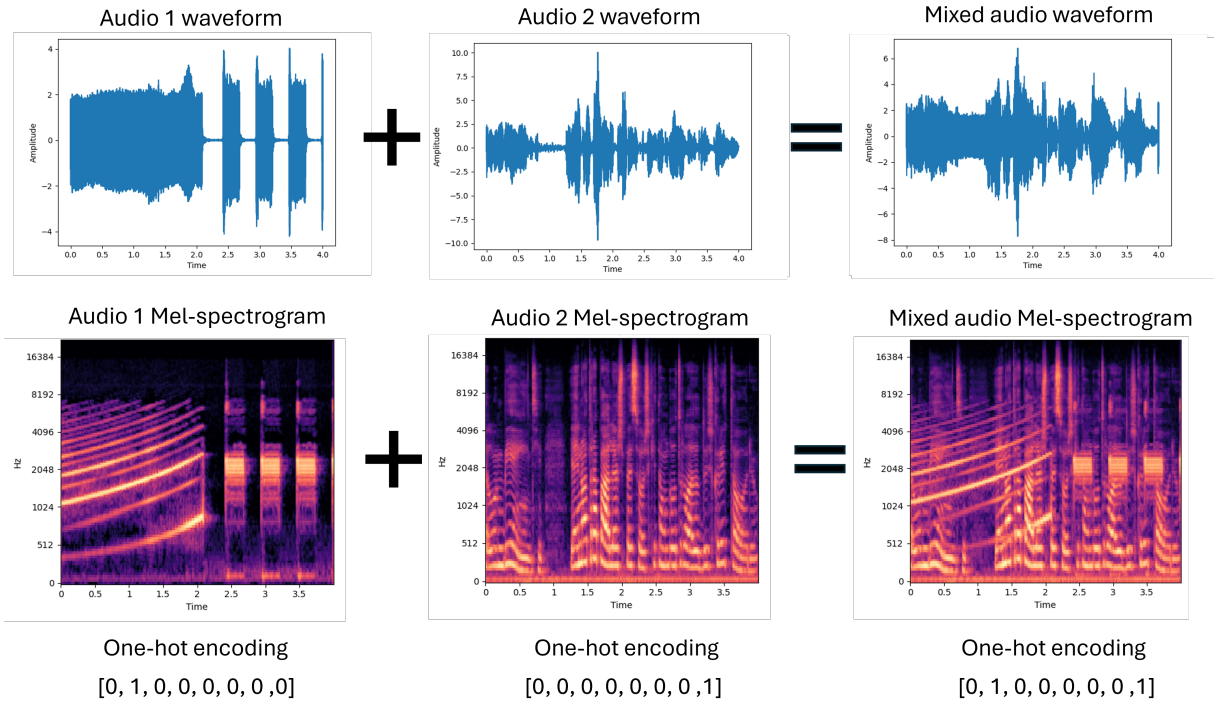


Figure 4.3 – Audio mixing scheme example.

Table 4.14 – Accuracy of the experiments using the CNN + RNN architecture and data augmentation.

Configuration	Accuracy without DA	Accuracy with DA	T-test
5	$90.7 \pm 1.49\%$	$90.4 \pm 1.26\%$	$(t(8) = 0.226, p = 0.827)$
6	$90.1 \pm 1.09\%$	$90.0 \pm 1.31\%$	$(t(8) = 0.140, p = 0.892)$
7	$91.3 \pm 1.15\%$	$91.9 \pm 0.84\%$	$(t(8) = 0.860, p = 0.415)$
8	$91.5 \pm 0.95\%$	$90.7 \pm 1.16\%$	$(t(8) = 1.020, p = 0.338)$

ularization and accuracy on various datasets, they did not prove effective for the dataset used in this work when applied with the proposed signal processing method. Similar findings are reported in the literature. Piczak (2015a) used random combinations of time shifting, pitch shifting, and time stretching for data augmentation, reporting that "simple augmentation techniques proved to be unsatisfactory for the UrbanSound8K dataset due to the considerable increase in training time they generated and their negligible impact on model accuracy". Another study by Turskis *et al.* (2023) used data augmentation on the ESC-50 and UrbanSound8K datasets. The results showed an improvement in the ESC-50 dataset, while the UrbanSound8K dataset did not benefit and, in some cases, even performed worse for both datasets.

4.6 Hierarchical CRNN results

A hierarchical classification approach was implemented in an attempt to increase the performance in classes of lower frequencies. A "Local classifier per parent Node"

model was chosen, which involves training a multi-class classifier for each parent node to distinguish among its child nodes. The spectral centroid was chosen as the criteria for dividing the dataset since it helps to define where most of the energy is localized in the spectrum (GIANNAKOPOULOS; PIKRAKIS, 2014). For this reason, the spectral centroid of each audio was calculated, and the average for each class is shown in Table 4.15.

Table 4.15 – List of classes ordered by spectral centroid.

Class	Spectral centroid (Hz)
Background noise	126
Heavy vehicle (idling)	199
Heavy vehicle (pass-by)	312
Car	313
Motorcycle	530
Helicopter	529
Air conditioner	774
Music	789
Voice	821
Siren	936
Dog	1018
Construction noise (impact)	1018
Alarm	1761
Rain	1996
Construction noise (non-impact)	2335
Bird	3337
Insect	3479

According to the centroid results, it was chosen that from "Background noise" to "Air conditioner" would be a group named "Low-frequency" and from "Music" to "Insect" would be a "High-frequency" group. The division between these groups occurs between "Air conditioner" and "Music". This split was chosen due to the similarity of these sound sources with their respective groups. "Air conditioner" shares more characteristics with other machine-like sounds, such as vehicles, typically in the low-frequency range. Conversely, "Music" resembles sounds like voices and dogs, which often include complex patterns and modulations characteristic of higher-frequency sounds.

Under the light of these two groups, the hierarchical classification proceeds. Firstly, a binary classifier identifies if the sample corresponds to the "Low-frequency" or "High-frequency" groups. This model is named Model 1. Afterward, depending on the result of the first classifier, the "Low-frequency" or "High-frequency" models are activated, and both multi-class classifiers decide which is the final class the sample belongs to. These models are named Model 1.1 and 1.2, respectively. Figure 4.4 depicts a schematic representation of the used hierarchy.

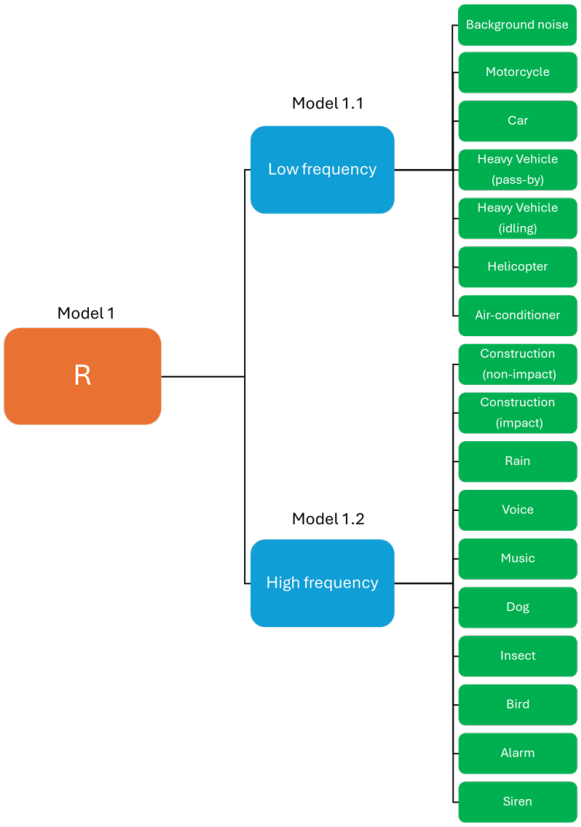


Figure 4.4 – Schematic representation of the hierarchical classification model.

The architectures used for Model 1, 1.1, and 1.2 are the same CRNN model presented in Section 3.4.4, the only difference is the output layer where Model 1 has 2 nodes, Model 1.1 has 7 nodes and Model 1.2 has 10 nodes, so they can adapt to the correspondent number of classes. Table 4.16 shows the accuracy comparison between the CRNN and the hierarchical models, using configurations 5, 6, 7, and 8, with the respective standard deviations. The two-tailed t-test is presented in the same table, comparing the original CRNN model and the new model using a hierarchical structure.

Table 4.16 – Accuracy comparison between the CRNN and the hierarchical model.

Configuration	CRNN model acc.	Hierarchical model acc.	T-test
5	$90.7 \pm 1.49\%$	$91.1 \pm 0.84\%$	$(t(8) = 0.516, p = 0.620)$
6	$90.1 \pm 1.09\%$	$91.1 \pm 0.71\%$	$(t(8) = 1.484, p = 0.176)$
7	$91.3 \pm 1.15\%$	$91.7 \pm 0.89\%$	$(t(8) = 0.586, p = 0.574)$
8	$91.5 \pm 0.95\%$	$90.8 \pm 1.00\%$	$(t(8) = 0.986, p = 0.353)$

Models 5, 6, and 7 presented a rise in accuracy, and Model 8 a decrease. However, the t-test also clarifies that the accuracy difference is not significant. Despite being statically equivalent, the hierarchical model can be considered less efficient, requiring much more computational effort than the smaller CRNN model.

A class-wise precision comparison between the hierarchical model and the CRNN can be made using tables 4.17 and 4.11. Overall, it is not possible to point out

a significant difference between the models. Therefore, the confusion at low frequencies persists at similar levels, even when using a specialized model for this frequency range.

Table 4.17 – Precision resulted by the hierarchical model for each configuration.

	Precision (%)			
Configuration	5	6	7	8
Bkg noise	84.2 ±3.8	**84.5** ±5.2	82.1 ±1.7	81.9 ±4.4
Motorcycle	78.6 ±4.3	78.0 ±4.8	77.9 ±6.1	**86.3** ±5.8
Car	**87.1** ±1.9	82.8 ±3.8	85.8 ±5.8	81.6 ±4.1
HV (pass-by)	70.4 ±4.5	**74.1** ±3.3	73.8 ±6.1	69.5 ±5.8
HV (idling)	92.9 ±3.3	**95.7** ±2.4	95.3 ±2.9	93.6 ±3.2
Helicopter	87.6 ±2.6	86.5 ±4.1	**91.2** ±4.3	85.6 ±6.1
Air conditioner	88.4 ±3.6	91.5 ±1.9	**93.0** ±5.0	88.6 ±1.4
Const (non-imp)	96.4 ±1.9	96.9 ±2.5	**98.3** ±1.8	96.4 ±2.5
Const (imp)	**97.8** ±1.6	**97.8** ±1.3	97.4 ±1.9	97.1 ±1.2
Rain	**97.7** ±1.6	95.7 ±3.8	97.4 ±1.7	95.8 ±2.6
Voice	95.6 ±2.3	94.9 ±2.4	**96.5** ±1.8	94.9 ±2.6
Music	95.7 ±2.9	93.5 ±2.7	93.5 ±2.3	**96.0** ±1.7
Dog	93.8 ±2.2	95.4 ±2.1	**95.5** ±1.5	95.4 ±1.6
Insect	**98.4** ±1.5	97.0 ±1.9	97.7 ±1.3	97.4 ±2.1
Bird	96.4 ±2.8	97.2 ±0.6	**97.5** ±1.3	96.8 ±1.7
Alarm	97.9 ±2.0	98.5 ±1.9	98.2 ±1.8	**99.1** ±0.8
Siren	94.7 ±2.7	92.9 ±4.3	**95.5** ±1.5	93.9 ±1.3
Macro avg.	91.4 ±0.7	91.3 ±0.5	**92.2** ±0.9	91.2 ±1.0
Weighted avg.	91.3 ±0.7	91.2 ±0.6	**92.0** ±0.8	91.0 ±1.0

5 Conclusion

This work explored the construction of an audio dataset tailored for the Brazilian soundscape in the city of São Paulo. Almost six thousand audio sample data distributed in 17 classes were collected comprising the main sound sources occurrences in the city, from natural sounds to machinery. Afterward, the sound classification problem using this dataset was tackled using machine learning and neural network architectures: linear SVM, CNN, and RNN. As input, 8 configurations of different Log-Mel spectrogram resolutions were processed.

The linear SVM model has clearly shown poorer performance than the neural networks, proving that a non-linear model is necessary for this type of task. The RNN (LSTM) model is a good improvement compared to a linear SVM for all resolutions of Log-Mel spectrograms. In CNN, increasing the resolution did not mean an accuracy improvement, considering that Configuration 3 performed best. The RNN (LSTM) model performed worse than the CNN model in four of the configurations. However, the RNN and CNN models exhibited similar performance in the remaining four configurations. These results are consistent with the literature, where the CNN architecture is frequently the most commonly used for sound classification tasks due to its performance.

A hybrid model approach using both CNN and RNN (LSTM) layers has shown to be an important evolution for the sound classification task of this work. The CRNN model has outperformed in all configurations over the previous CNN and RNN architectures. Moreover, there is a clear trend of growth in accuracy when increasing the Log-Mel spectrogram resolution, proving that capturing both spatial features and time relationships within samples is essential for classification efficiency.

A class-wise analysis of the results showed similar observations for all models: a difficulty in classifying the low-frequency machinery sound sources, especially vehicles, as depicted in precision, recall, and f1-score. This is explained by the similarity of spectral components and how they are generated. “Heavy vehicles (pass-by)” and “Motorcycle” presented the worst results, in general. In contrast, high-frequency sounds of this dataset contain much more unique patterns and are easily distinguishable, resulting in scores near 99% in classes such as “Alarm” and “Insect”.

The data augmentation techniques applied in this work have not shown significant evolution in the results for the dataset under study compared to the CRNN. Despite being an important approach for regularization, preventing overfitting, and improving the accuracy of a model, these results are not atypical and are occasionally described in the

literature. So, it is important to be aware that this technique may not always guarantee a better performance.

The last model improvement used a hierarchical classification, splitting the dataset by its spectral centroid into two groups: low frequency and high frequency. The idea of having a specialized network trained for each group has not shown improvement compared to the CRNN results. Even for a class-wise comparison, the low-frequency problem could not be sorted out using this approach, leading to the same errors throughout the classes. The hierarchical model used in this work has not proven viable since it results in the same performance using much more computational power.

5.1 Future work

Future research could further explore and enhance the findings of this study. Some steps to be investigated in future research of this work include:

- **Complete the full dataset:** Expand the dataset by including the missing classes and reevaluating the performance of the models could provide a more comprehensive understanding of models' capabilities and address gaps in the current dataset.
- **Real-world applications:** Analyze the performance for real-world applications embedded in sound monitoring stations, so a huge amount of new unseen data is gathered in several different conditions.
- **Real-time classification:** Investigate the performance of models for real-time classification and their cost-benefit considering power consumption and performance.
- **Hybrid and ensemble models:** Continue to develop and refine hybrid models, such as combinations of CNN and RNN layers. Examples include dual stream input for CNN and RNN, for instance (BAE *et al.*, 2016). Additionally, investigate the use of ensemble methods to potentially improve classification accuracy by combining the strengths of multiple models.
- **Multi-features input:** Use other features simultaneously as input, such as waveforms and MFCCs (Mel-Frequency Cepstral Coefficients) to provide diverse representations of the audio data and enhance the model's ability to capture different aspects of the sound.
- **Human-model comparison:** Further investigate the comparison between human and model performance in distinguishing sounds. Some studies have shown that humans also struggle to differentiate certain sounds that models find challenging

(ASHHAD *et al.*, 2023). Since models usually mimic the human hearing system, it is natural they encounter similar difficulties, therefore, analyzing these similarities could provide more insights about the models' performances, and exploring architectures beyond human capacity may be necessary.

Bibliography

ABESSER, J.; GOURISHETTI, S.; KÁTAI, A.; CLAUSS, T.; SHARMA, P.; LIEBETRAU, J. **IDMT-Traffic: An Open Benchmark Dataset for Acoustic Traffic Monitoring Research**. *European Signal Processing Conference*, v. 2021-Augus, p. 551–555, 2021. ISSN 22195491. Cited on page 44.

ACHARYA, J.; BASU, A. **Deep Neural Network for Respiratory Sound Classification in Wearable Devices Enabled by Patient Specific Model Tuning**. *IEEE Transactions on Biomedical Circuits and Systems*, v. 14, n. 3, p. 535–544, 2020. ISSN 19409990. Cited on page 15.

ADAVANNE, S. ***Sound event localization, detection, and tracking by deep neural networks***. [S.l.: s.n.], 2020. ISBN 9789520314613. Cited on page 25.

ALJUBAYRI, I. **Comparative Analysis of Different Sampling Rates on Environmental Sound Classification Using the Urbansound8k Dataset**. *Journal of Computer and Communications*, v. 11, n. 06, p. 19–27, 2023. ISSN 2327-5219. Cited on page 52.

ARIAS-VERGARA, T.; KLUMPP, P.; VASQUEZ-CORREA, J. C.; NÖTH, E.; OROZCO-ARROYAVE, J. R.; SCHUSTER, M. **Multi-channel spectrograms for speech processing applications using deep learning methods**. *Pattern Analysis and Applications*, Springer London, v. 24, n. 2, p. 423–431, 2021. ISSN 1433755X. Disponível em: <https://doi.org/10.1007/s10044-020-00921-5>. Cited on page 40.

ASHHAD, M.; GOENKA, U.; JAGETIA, A.; AKHTARI, P.; AMBAT, S. K.; SAMUEL, M. **Improved Vehicle Sub-type Classification for Acoustic Traffic Monitoring**. *2023 National Conference on Communications, NCC 2023*, 2023. Cited 2 times on pages 64 and 74.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). ***ABNT NBR 10151**: Acústica - Medição e avaliação de níveis de pressão sonora em áreas habitadas - Aplicação de uso geral*. 2019. Rio de Janeiro, Brasil. Cited on page 46.

BAE, S. H.; CHOI, I.; KIM, N. S. **Acoustic scene classification using parallel combination of LSTM and CNN**. In: . [S.l.: s.n.], 2016. Cited on page 73.

BIANCO, M. J.; GERSTOFT, P.; TRAER, J.; OZANICH, E.; ROCH, M. A.; GANNOT, S.; DELEDALLE, C.-A. **Machine learning in acoustics: Theory and applications**. *The Journal of the Acoustical Society of America*, Acoustical Society of America, v. 146, n. 5, p. 3590–3628, 2019. ISSN 0001-4966. Cited 2 times on pages 17 and 22.

BODDAPATI, V. **Classifying Environmental Sounds with Image Networks**. *Thesis*, n. February, p. 37, 2017. Disponível em: <www.bth.se>. Cited on page 52.

BODDAPATI, V.; PETEF, A.; RASMUSSON, J.; LUNDBERG, L. **Classifying environmental sounds using image recognition networks**. *Procedia Computer Science*, Elsevier B.V., v. 112, p. 2048–2056, 2017. ISSN 18770509. Disponível em: <http://dx.doi.org/10.1016/j.procs.2017.08.250>. Cited on page 52.

BORIN, M.; MASIERO, B.; MONTEIRO, C. **Exploring deep learning architectures for urban sound classification**. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, v. 268, n. 3, p. 5786–5796, 2023. ISSN 0736-2935. Cited on page 46.

CAKIR, E. ***Deep Neural Networks for Sound Event Detection***. [s.n.], 2019. v. 12. 129 p. ISBN 9789520309619. Disponível em: <http://urn.fi/URN:ISBN:978-952-03-0962-6>. Cited 3 times on pages 25, 32, and 36.

CAKIR, E.; HEITTOLA, T.; HUTTUNEN, H.; VIRTANEN, T. **Polyphonic sound event detection using multi label deep neural networks**. *Proceedings of the International Joint Conference on Neural Networks*, v. 2015-Septe, n. July, 2015. Cited 3 times on pages 7, 38, and 39.

CARTWRIGHT, M.; MENDEZ, A. E. M.; CRAMER, J.; LOSTANLEN, V.; DOVE, G.; WU, H.-H.; SALAMON, J.; NOV, O.; BELLO, J. **SONYC Urban Sound Tagging (SONYC-UST): A Multilabel Dataset from an Urban Acoustic Sensor Network**. *Detection and Classification of Acoustic Scenes and Events 2019*, n. October, p. 35–39, 2019. Cited on page 43.

CHU, S.; NARAYANAN, S.; KUO, C. C. **Environmental sound recognition with Time-Frequency audio features**. *IEEE Transactions on Audio, Speech and Language Processing*, v. 17, n. 6, p. 1142–1158, 2009. ISSN 15587916. Cited 2 times on pages 44 and 48.

COCKOS INCORPORATED. ***Reaper: Digital Audio Workstation***. <https://www.reaper.fm/>. Accessed on February 7, 2024. Cited on page 47.

EUROPEAN ENVIRONMENT AGENCY. ***Health risks caused by environmental noise in Europe***. [S.l.]: Publications Office, 2020. Cited on page 13.

FONT, F.; ROMA, G.; SERRA, X. **Freesound technical demo**. *Proceedings of the 2013 ACM Multimedia Conference*, p. 411–412, 2013. Disponível em: <https://dl.acm.org/doi/10.1145/2502081.2502245>. Cited on page 15.

G1 SP; TV GLOBO. ***Reclamações de barulho ao Psiu crescem 41% na cidade de SP no 1 semestre de 2023***. 2023. Disponível em: <https://g1.globo.com/sp/sao-paulo/noticia/2023/07/27/reclamacoes-de-barulho-ao-psiu-crescem-41percent-na-cidade-de-sp-no-1o-semestre-de-2023.ghtml>. Cited on page 13.

GALILEU, J. P. D. **Urban Sound Event Classification for Audio-Based Surveillance Systems**. n. January, p. 64, 2020. Cited 2 times on pages 7 and 39.

GENÇAY, R.; QI, M. **Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging**. *IEEE Transactions on Neural Networks*, v. 12, n. 4, p. 726–734, 2001. ISSN 10459227. Cited 3 times on pages 7, 32, and 33.

GERON, A. ***Hands–On Machine Learning with Scikit–Learn and TensorFlow 2nd edition***. [S.l.: s.n.], 2019. 838 p. ISBN 9781492032649. Cited on page 18.

GIANNAKOPOULOS, T.; PIKRAKIS, A. **Chapter 4 - Audio Features**. In: GIANNAKOPOULOS, T.; PIKRAKIS, A. (Ed.). *Introduction to Audio Analysis*. Oxford: Academic Press, 2014. p. 59–103. ISBN 978-0-08-099388-1. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780080993881000042>. Cited on page 69.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning: Machine Learning Book**. p. 785, 2016. Disponível em: <http://www.deeplearningbook.org/>. Cited on page 32.

GUO, J.; LI, C.; SUN, Z.; LI, J.; WANG, P. **A Deep Attention Model for Environmental Sound Classification from Multi-Feature Data**. *Applied Sciences (Switzerland)*, v. 12, n. 12, 2022. ISSN 20763417. Cited on page 52.

GUSTINELI, M. **A survey on recently proposed activation functions for Deep Learning**. p. 1–7, 2022. Disponível em: <http://arxiv.org/abs/2204.02921>. Cited on page 26.

HÖGSKOLA, L. T.; LEXFORS, L.; JOHANSSON, M. **Audio representation for environmental sound classification using convolutional neural networks**. 2018. Cited on page 52.

JORDAL, I. ***iver56/audiomentations: v0.21.0***. Zenodo, 2022. Disponível em: <https://doi.org/10.5281/zenodo.6046340>. Cited on page 66.

KINGMA, D. P.; BA, J. ***Adam: A Method for Stochastic Optimization***. 2017. Disponível em: <https://arxiv.org/abs/1412.6980>. Cited on page 32.

KLOTH, M.; VANCLUYSEN, K.; CLEMENT, F.; ELLEBJERG, P. L.; DRI. ***Practitioner Handbook for Local Noise Action Plans: Recommendations from the SILENCE project***. [s.n.], 2008. 124 p. Disponível em: <http://www.silence-ip.org/site/fileadmin/SP_J/E-learning/Planners/SILENCE_Handbook_Local_noise_action_plans.pdf>. Cited on page 13.

KOUTINI, K. ***Large-Scale Weakly Supervised Sound Event Detection For Smart Cars***. Tese (Doutorado) — Johannes Kepler University Linz - JKU, 2018. Cited on page 32.

LYON, R. F. **Human and Machine Hearing**. *Human and Machine Hearing*, 2017. Cited on page 15.

MCFEE *et al.* ***Librosa: 0.8.1***. Zenodo, 2021. Disponível em: <https://doi.org/10.5281/zenodo.8252662>. Cited on page 52.

MOTA; CAMILLA, V. ***Demolições em alta apagam memória de bairros tradicionais de São Paulo***. 2021. Disponível em: <https://www1.folha.uol.com.br/cotidiano/2021/10/demolicoes-em-alta-apagam-memoria-de-bairros-tradicionais-de-sao-paulo.shtml>. Cited on page 13.

MU, W.; YIN, B.; HUANG, X.; XU, J.; DU, Z. **Environmental sound classification using temporal-frequency attention based convolutional neural network**. *Scientific Reports*, Nature Publishing Group UK, v. 11, n. 1, p. 1–14, 2021. ISSN

20452322. Disponível em: <https://doi.org/10.1038/s41598-021-01045-4>. Cited on page 14.

MYDLARZ, C.; SHARMA, M.; LOCKERMAN, Y.; STEERS, B.; SILVA, C.; BELLO, J. P. **The life of a new york city noise sensor network**. *Sensors (Switzerland)*, v. 19, n. 6, p. 1–24, 2019. ISSN 14248220. Cited on page 43.

NASIRI, A. ***Deep Learning Based Sound Event Detection and Classification***. 1–114 p. Tese (Doutorado) — University of South Carolina, 2021. Disponível em: <https://scholarcommons.sc.edu/etd>. Cited 2 times on pages 7 and 38.

PAJANKAR, A.; JOSHI, A. ***Introduction to Machine Learning with Scikit-learn***. [S.l.: s.n.], 2022. 65–77 p. ISBN 9781449369415. Cited 2 times on pages 18 and 37.

PICZAK, K. J. **Environmental sound classification with convolutional neural networks**. In: . [S.l.: s.n.], 2015. ISBN 978-1-4673-7454-5. Cited on page 68.

PICZAK, K. J. **ESC: Dataset for environmental sound classification**. *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, p. 1015–1018, 2015. Cited on page 45.

PUENTE, S. A.-B. ***Single and Multi-Label Environmental Sound Classification Using Convolutional Neural Networks***. 1–62 p. Tese (Doutorado) — Chalmers University of Technology, 2018. Cited on page 25.

RITCHIE, H.; ROSER, M. ***Urbanization***. 2018. <https://ourworldindata.org/urbanization>. Accessed: 2024-08-01. Cited on page 13.

ROCH, M. A. **How Machine Learning Contributes to Solve Acoustical Problems**. *Acoustics Today*, v. 17, n. 4, p. 48, 2021. ISSN 15570215. Cited 2 times on pages 34 and 55.

ROSEBROCK, A. ***Deep Learning for Computer Vision with Python - Practitioner Bundle***. [S.l.: s.n.], 2017. Cited on page 33.

ROSEBROCK, A. ***Deep Learning for Computer Vision with Python - Starter Bundle***. 1. ed. [S.l.]: PYIMAGESEARCH, 2017. 330 p. Cited 3 times on pages 15, 23, and 34.

RUDER, S. **An overview of gradient descent optimization algorithms**. p. 1–14, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>. Cited on page 31.

SALAMON, J.; BELLO, J. P. **Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification**. *IEEE Signal Processing Letters*, v. 24, n. 3, p. 279–283, March 2017. ISSN 1558-2361. Cited on page 40.

SALAMON, J.; JACOBY, C.; BELLO, J. P. **A Dataset and Taxonomy for Urban Sound Research**. *Proceedings of the 22nd ACM international conference on Multimedia*, p. 1041–1044, 2014. Disponível em: <http://dx.doi.org/10.1145/2647868.2655045.> Cited 3 times on pages 14, 44, and 48.

SCHMIDT, R. M. **Recurrent Neural Networks (RNNs): A gentle Introduction and Overview**. n. 1, p. 1–16, 2019. Disponível em: <http://arxiv.org/abs/1912.05911>. Cited 2 times on pages 7 and 35.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. **Activation Functions in Neural Networks**. *International Journal of Engineering Applied Sciences and Technology*, v. 04, n. 12, p. 310–316, 2020. Cited on page 26.

SILLA, C. N.; FREITAS, A. A. **A survey of hierarchical classification across different application domains**. *Data Mining and Knowledge Discovery*, v. 22, n. 1-2, p. 31–72, 2011. ISSN 13845810. Cited 4 times on pages 7, 19, 20, and 21.

SÃO PAULO. ***LEI Nº 16.499, DE 20 DE JULHO DE 2016***. 2016. <https://www.legislacao.sp.gov.br/legislacao/leis/16499>. Disponível em: <https://www.legislacao.sp.gov.br/legislacao/leis/16499>. Cited on page 13.

SÃO PAULO. 2021. Disponível em: <http://dados.prefeitura.sp.gov.br/>. Cited on page 13.

SÃO PAULO. ***DECRETO Nº 60.581, DE 27 DE SETEMBRO DE 2021. Regulamenta o controle de ruídos na execução das obras de construção civil no Município de São Paulo.*** 2021. Cited on page 13.

TERVEN, J.; CORDOVA-ESPARZA, D. M.; RAMIREZ-PEDRAZA, A.; CHAVEZ-URBIOLA, E. A. **Loss Functions and Metrics in Deep Learning. A Review**. p. 1–53, 2023. Disponível em: <http://arxiv.org/abs/2307.02694>. Cited on page 29.

TURSKIS, T.; TELEISA, M.; BUCKIUNAITE, R.; ČALNERYTE, D. **Mixed-Type Data Augmentations for Environmental Sound Classification**. In: . [S.l.: s.n.], 2023. p. 184–194. Cited on page 68.

VEMURI, V. K. **The Hundred-Page Machine Learning Book**. *Journal of Information Technology Case and Application Research*, v. 22, n. 2, p. 136–138, 2020. ISSN 1522-8053. Cited on page 17.

VIDAÑA-VILA, E.; DUBOC, L.; ALSINA-PAGÈS, R. M.; POLLS, F.; VARGAS, H. **BCNDataset: Description and analysis of an annotated night urban leisure sound dataset**. *Sustainability (Switzerland)*, v. 12, n. 19, 2020. ISSN 20711050. Cited on page 46.

VIDAÑA-VILA, E.; NAVARRO, J.; STOWELL, D.; ALSINA-PAGèS, R. M. **Multilabel acoustic event classification using real-world urban data and physical redundancy of sensors**. *Sensors*, MDPI, v. 21, 11 2021. ISSN 14248220. Cited on page 67.

WANG, J. C.; WANG, J. F.; LIN, C. B.; JIAN, K. T.; KUOK, W. H. **Content-based audio classification using support vector machines and independent component analysis**. *Proceedings - International Conference on Pattern Recognition*, v. 4, n. January 2006, p. 157–160, 2006. ISSN 10514651. Cited on page 24.

YUSNITA, M. A.; PAULRAJ, M. P.; YAACOB, S.; YUSUF, R.; SHAHRIMAN, A. B. **Analysis of accent-sensitive words in multi-resolution mel-frequency cepstral coefficients for classification of accents in Malaysian english**. *International Journal of Automotive and Mechanical Engineering*, v. 7, n. 1, p. 1053–1073, 2013. ISSN 21801606. Cited 2 times on pages 7 and 40.

ZHANG, Z.; XU, S.; ZHANG, S.; QIAO, T.; CAO, S. **Attention based convolutional recurrent neural network for environmental sound classification**. *Neurocomputing*, v. 453, p. 896–903, 2021. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231220313618>. Cited on page 14.

ZOONIVERSE. ***Zooniverse: People-powered research***. 2024. <https://www.zooniverse.org/>. Accessed: 2024-08-04. Cited on page 43.