Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Ciencias en Inteligencia Artificial

CLASSIFICATION OF MULTIPLE SOUND EVENTS IN A SINGLE FRAME USING GENERATIVE ADVERSARIAL NETWORKS

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias en Inteligencia Artificial

Presenta:

Rodrigo González Huerta

Dirigido por:

Dr. Juan Manuel Ramos Arreguín

SINODALES

Dr. Juan Manuel Ramos Arreguín

Presidente

Dr. Andras Takacs

Secretario

Dr. Edgar Alejandro Rivas Araiza

Vocal

Dr. Saúl Tovar Arriaga

Suplente

Dr. Marco Antonio Aceves Fernández

Suplente

Centro Universitario, Querétaro, Querétaro, México;

Junio 2023

Classification of multiple sound events in a single frame using generative adversarial networks

por

Rodrigo González Huerta

Para Emiliano, Cathia y Eva.

Acknowledgments

Agradezco al Dr. Andras Takacs por su tutoría durante el desarrollo de este trabajo, al Dr. Juan Manuel Ramos Arreguín por el apoyo brindado, al Dr. Marco Antonio Aceves Fernández, Dr. Saúl Tovar Arriaga y Dr. Edgar Alejandro Rivas Araiza por ser parte de mi sínodo y leer este trabajo. Finalmente, agradezco al Mtro. Angel Balderas Paredes por los consejos y la ayuda brindada.

Abstract

Environmental sound classification is a computational task that belongs to the branch of artificial intelligence called sound recognition. Several techniques and different approaches exist to tackle this task; one that yields excellent results is through the utilization of deep learning techniques, i.e., neural networks. Despite their good results, neural networks in some cases fail to generalize well to new data when the amount of training data is scarce. This can lead to a phenomena called overfitting. A solution to this inconvenience is based on the use of deep generative models to generate synthetic data through the approximation of high-dimensional probability distributions. This allow to generate new samples, similar to the ones used to train the generative model. Generative Adversarial Networks (GANs) are a kind of generative model which trains two neural networks simultaneously in an adversarial way, i.e., pitting one against the other. In this work it is shown the effect of using GANs as data augmentation technique that could be used to improve the performance of different sound classification models.

Keywords: Neural networks, environmental sounds, data augmentation.

Resumen

La clasificación de sonidos ambientales es una tarea computacional perteneciente a la rama de la inteligencia artificial llamada reconocimiento de sonido. Existen varias técnicas para abordar esta tarea; una que arroja excelentes resultados es mediante la utilización de técnicas de aprendizaje profundo, es decir, redes neuronales. A pesar de sus buenos resultados, las redes neuronales en algunos casos no logran generalizar bien a nuevos datos cuando la cantidad de datos de entrenamiento es escasa. Esto puede conducir a un fenómeno llamado sobreajuste. Una solución a este inconveniente se basa en la utilización de modelos generativos profundos para generar datos sintéticos a través de la aproximación de distribuciones de probabilidad de alta dimensionalidad. Esto permite generar nuevas muestras, similares a las utilizadas para entrenar el modelo generativo. Las redes generativas antagónicas (GANs) son un tipo de modelo generativo que entrena dos redes neuronales simultáneamente de forma antagónica, es decir, enfrentando una contra la otra. En este trabajo se muestra el efecto del uso de GANs como técnica de aumento de datos que pudiera servir para mejorar el rendimiento de diferentes modelos de clasificación de sonido.

Palabras clave: Redes neuronales, sonidos ambientales, aumento de datos.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter will give an introduction to the background of the subject of this thesis, which is the classification of environmental sounds. The chapter includes what is the motivation for this work, the main obstacles or difficulties and a review of the literature. At the end of the chapter, the hypothesis and the objectives are presented with the intention of giving the reader a better sense of what was intended to be achieved with this work.

1.1 Background

Nowadays, Artificial Intelligence (AI) along with other technologies pertaining to the field of Computer Science are being incorporated in a natural manner into businesses and people's daily lives. Some examples are big data and the internet of things (IoT). The reason why society is rapidly adopting Artificial Intelligence has to do with its enormous potential in building systems that perform extremely well on tasks that for humans would take a long time to be done or simply are hard enough to even try to accomplish them. Even in simple tasks like classifying sounds or images humans are being surpassed by some of this algorithms.

The sound domain is one of the areas that Artificial Intelligence has been increasingly exploring over the last decades. Particularly, speech recognition applications had become popular and some of them are now available in a large number of devices, especially in cell phones. In general, when dealing with sound, the type of output a system computes depends largely on the application one wishes to program. For example, a subject that has been studied lately is Environmental Sound Classification (ESC), a single-label classification task whose

purpose is to assign the correct label from a set of predefined categories to each sound file from a dataset. Furthermore, Environmental Sound Classification can be expanded into a multi-label classification task, so that a system becomes capable of recognizing more than one class of sound present in an audio frame. This application, also known as Audio Tagging, in practice would be more useful since normally environmental audio signals are composed of various classes of sounds, some of them in the foreground and others in the background. Audio Tagging could also be expanded so that an application not just recognizes the classes of sounds present in an audio but also the environments in which they are taking place, for example, a house, an airport or the street (Audio Scene Classification). Another ramification of this problem consists in identifying the temporal lapses in which sounds are active (Sound Event Detection). More specific tasks deal with estimating the distance from a microphone to the source of sound, or determine if two audio recordings come from the same sound source [1].

Deep Learning algorithms, i.e., artificial neural networks, have been proved to perform well in classification tasks such as image classification and recently, it has been shown they also work well solving problems that deal with sound; particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are applicable algorithms for classification of monophonic and polyphonic sound datasets [2, 3, 4]. The reason of this is because artificial neural networks are algorithms capable of handling huge amounts of data with which they are able to model complex patterns from which, depending on the problem in hand, a desired output could be extracted. However, the success of all learning algorithms like artificial neural networks and other machine learning models depends largely on the amount of data (samples) available for encompass the high variability that certain types of data exhibit. Some databases that could be considered large actually wouldn't allow to train a learning model enough so that it could exhibit the desired behaviour during the performance of the task for which it has been designed. To make matters worse, acquiring additional samples is sometimes too expensive or simply not feasible to do.

There are ways to attack this drawback without the need of acquiring new samples. Data augmentation refers to a set of techniques that addresses this problem by generating synthetic data from the samples available in a dataset. Some data augmentation methods that work well in other domains, for example in image processing, have adapted well to the domain of sound. An example of this is the addition of random noise. Other basic transformations for sound include pitch shifting, dynamic range compression and time stretching.

Although the development of data augmentation techniques isn't something new, until recently computer scientists began to look for solutions of data scarcity using deep learning models. Generative adversarial networks (GANs) belong to a special kind of generative algorithms, known as generative deep learning algorithms, whose function in essence consists of training simultaneously two neural networks in an adversarial way [5]. GANs are being used to deal with the issue of data scarcity and they are giving promising results [6]. However, despite its usefulness, which has already been demonstrated in applications for the area of image processing, the use of this technology for the domain of sound is still at an early stage.

This thesis is about the implementation of audio classification systems for both monophonic (single-label) and polyphonic (multi-label) environmental sound datasets. Furthermore, this work covers a study of how the use of a specific type of generative adversarial network called WGAN-GP, a novel algorithm that has not been widely studied for the area of sound, enhances the robustness of this classifiers by increasing the size of one dataset. Different evaluation metrics were used to assess the performance of the models.

1.2 Motivation

Environmental sounds are encountered in everyday life; they encompass most classes of sounds, including human-made sounds and non-human made sounds, from the ones produced in natural environments to those that fill a city. It could be said that all sounds belong to this categorization with the only exception of music and spoken language. However, despise its almost overwhelming availability, creating machine learning systems that could automatically classify environmental sounds is an open problem for exploration and a challenging one, although the study of this problem is not new. Different methods and methodologies have been proposed, however, it is a subject in which contributions can still be made.

On the other hand, facing the challenge is very appealing because environmental sound recognition is an area of research with enormous potential for creating applications that could bring benefits to society; this kind of applications could be used to detect and categorize the sources of noise pollution that permeate a city [7]. This is an important issue that is gaining a lot of attention because of the rapid increment in the size of urban areas and the continuous exposure to prolonged or excessive noises millions of people experience every day which in the long run could bring health issues, from stress and sleep disturbances to high blood pressure, heart diseases and noise induced hearing loss [8].

Environmental sound recognition systems could also improve surveillance systems, allowing to recognize particular events like gunshots, glass breaking or the barking of a guardian dog [9]. Other applications with potential include home automation and the generation of quicker responses in case of accidents, for example, car crashes or explosions. People that are in some way dependent of assistance like elders or people with certain kinds of disabilities like deafness or blindness, could also benefit from this novel technology.

1.3 Problem Formulation

Single-label and multi-label audio classification tasks are gaining a lot of attention recently; this attention come from the difficulty of designing accurate systems given the enormous variability that almost any class of sound exhibit under uncontrolled conditions. For environmental sounds, complexity is higher; the lack of recognizable patterns makes it seem like there is no clear way to attack the problem, unlike other type of sounds, e.g. speech, where phonetic structures consisting of identifiable basic units can be found [1].

Well performing systems would have to deal with a great variety of scenarios. The most common would be to find audio signals consisting of more than one foreground sound plus some background noise. This could turn a signal confusing even for humans. Variations in data from the same sound source would be common in real world applications, making almost necessary to build models able to deal efficiently with the variability of data.

A way to address this inconvenience would be to teach a learning system through large quantities of examples pertaining to each class of sound. However, the scarce number of available public datasets, which by the way are almost limited to urban sounds, brings the drawback that researchers in many cases would have to build their own datasets. The consequence of this is that many researches studying the same subject wouldn't be comparable between them and results couldn't be repeated. Added to this, the insufficient number of audio samples that make up these datasets and the limited number of classes restrict these type of systems to be efficient in a narrow range of environments.

Data Augmentation encompass a variety of techniques which intend to solve this problem through the generation of synthetic data from the available samples of a dataset. Some data augmentation techniques of other domains, e.g. image processing, have been adapted to the sound domain allowing to enhance the robustness of some learning algorithms like artificial

neural networks. There are other techniques of data augmentation that have not been fully analysed; Generative Adversarial Networks are one of them. This new type of generative modelling algorithm seems could perform better than other methods, however, GANs suffer usually display instability during training [10].

1.4 Literature Review

Environmental sound classification systems based on artificial neural networks have gained a lot of attention in the last few years. Some works have reported good results using conventional machine learning classifiers like SVM or Decision Trees [11], however, when dealing with environmental sounds it has been observed that Deep Learning models can outperform classical models [12]. One of the earliest works in this matter used convolutional neural networks on three ESC public databases (UrbanSound8K, ESC-10 and ESC-50) [2]. The main objective of this work was to evaluate whether convolutional neural networks were applicable algorithms for this task. The results obtained from this model were comparable to the ones obtained from other state-of-the-art models at that time. Before the use of convolutional neural networks, more common Machine Learning algorithms like Gaussian mixture models (GMMs), support vector machines (SVMs) and hidden Markov models (HMMs) were used [13].

Various types of feature extraction techniques have been used in environmental sound classification systems in order to reduce dimensionality and also highlight meaningful information of the available data. For example, in one study Mel Frequency Cepstral Coefficients, Log-Mel Coefficients and Mel Spectrograms from different augmented datasets (UrbanSound8K, ESC-10 and ESC-50) were extracted and evaluated individually on a convolutional neural network, obtaining accuracies of 94.94%, 89.28%, and 95.37% for each of those datasets [14]. For increasing the size of the datasets they used data augmentation techniques such as pitch shifting, silence trimming, time stretching and white noise addition.

Recurrent Neural Networks (RNNs) have also been employed in environmental sound analysis because of its efficiency and flexibility when working with time-series signals such as digital audio [15]. RNNs are an excellent choice when it is important to take into account temporal relationships. Furthermore, hybrid systems can also be designed. For example, in a study a convolutional recurrent neural network was designed in order to take advantage of the benefits of CNNs and RNNs [16]. This work also studied the effect of adding attention

mechanisms to deal with irrelevant parts of the data.

Neural networks can also be used for different tasks besides classification. For example, convolutional neural networks have been used as method of data augmentation [17, 18]. Data augmentation has been observed to be a valuable step to be taken into account for sound classification systems. In these studies, a modern architecture of neural network called Generative Adversarial Network (GAN) consisting of two CNNs was used for generating new samples of data; In [18] the important aspect is that they proposed two architectures of GANs based on another generative adversarial network called Wasserstein GAN. The first one produces raw audio signals (WaveGAN) while the other one produces spectrogram images (SpecGAN). They didn't go further and test either of those GANs on some classification task. In [17], they used a different type of GAN called Weighted Cycle-Consistent Generative Adversarial Network to increase the size of various monophonic datasets to analyse if this action enhanced the performance of a Random Forest classifier. Generative adversarial networks have also been evaluated against basic audio transformations such as pitch shifting, time stretching, background noise and dynamic range compression. A study obtained a significant increase in accuracy over those methods [19].

For the case of multi-label classification (Audio Tagging), the use of artificial neural networks have also been explored [4, 20]. In a study, DNNs fed with spectral-domain features (Mel-band energies, log Mel-band energies and MFCCs) achieved an overall F1-score of 63.8% [20]. 2015). Their model was evaluated "with recordings from realistic everyday environments" against a model based on a Hidden Markov Model classifier. The neural network outperformed the Hidden Markov Model by a considerable margin. Another study used a convolutional neural network as classifier, testing it with a relatively small polyphonic dataset created from a monophonic dataset (UrbanSound8k), allowing to obtain values for precision, recall and F1-score of 70.56%, 58.37%, 63.89%, respectively [21].

Regarding the use of data augmentation techniques on environmental Audio Tagging, no papers were found in which the use of generative adversarial networks was mentioned, although it is not intended to imply that these works do not exist. Thus, there is a study opportunity to explore the influence that could have the use of GANs for this task.

1.5 Hypothesis

Increasing the size of a dataset of environmental sounds using an appropriate combination of generator and discriminator in a Generative Adversarial Network would enhance the performance in both single-label (monophonic) and multi-label (polyphonic) classification tasks, achieving similar evaluation metrics than the state-of-the-art methods.

1.6 Objective

Compare the performance of distinct classifiers designed for Environmental Sound Classification (single-label classification) and Audio Tagging (multi-label classification) using as input the features extracted from at least one public dataset increased in size by using different data augmentation techniques, including one type of Generative Adversarial Network.

1.6.1 Specific Objectives

- Propose and implement a sound processing method defining the filters and basic parameters for data cleaning and augmentation.
- Propose and implement the method to combine individual classes of sounds to create a polyphonic sound dataset from a monophonic one.
- Implement two or more methods of data augmentation, including one generative adversarial network, to create additional samples for each of the classes in the monophonic dataset.
- Implement at least two classifiers for each task and evaluate its performance with at least one dataset.
- Propose and implement the evaluation metrics, comparing the obtained results with the ones obtained by the state-of-the-art.

CHAPTER 2

Theoretical Background

In this chapter a study of the theory behind this research is presented. The main topics covered are artificial neural networks and audio signal processing. The first topic includes a description of each of the different models that were implemented to accomplish the tasks of classification and data augmentation. The other topic, audio signal processing, starts with a brief description of what is sound and how useful information can be extracted from it using different mathematical tools such as the Fourier transform, which ultimately will serve to perform the tasks needed to fulfil the objectives of this work.

2.1 Machine Learning

Machine learning is an area of Artificial Intelligence which deals with creating algorithms capable of extracting patterns from data without explicit human programmed instructions; in other words, algorithms that "learn". With this knowledge which could also be seen as experience, machine learning algorithms improve gradually their performance on a specific task, which means they start generalizing well to new (unseen) data. What's more, how well an algorithm learns can be measured by certain, sometimes task specific, performance measures.

Some of the reasons it is appealing to build systems that are able to learn from data is because programmers in general can't anticipate all the possible situations a system will face and because for many problems, implementing an algorithm that outputs a full solution is an arduous task, if not impossible. Different types of tasks can be solved through machine learning algorithms, being the most common tasks of classification, regression, synthesis of

new data, imputation of missing values, anomaly detection, denoising, etc.

2.1.1 Supervised Learning

Sometimes the output for a specific task is known beforehand or the samples witch which the system is fed are labelled (often manually). With this information a machine learning model is capable of learning the relation between the inputs and outputs more easily during the training phase. This is known as supervised learning. By learning distinguishing features of each class, belonging to the training set, supervised learning models can predict correct outcomes from unseen data. An example of a supervised learning task would be Environmental Sound Classification and Audio Tagging, because most datasets in addition of providing the audio recordings, they also provide indicators of which class of sound is present in those recordings.

If the data is not labelled, then a different approach must be taken. In unsupervised learning the target value is missing and it has to be inferred in some way.

2.1.2 Artificial Neural Networks

Artificial neural networks are part of a group of computational models that belong to the sub-area of machine learning called Deep Learning. This models can be categorized as supervised learning models since usually the correct output for every example used to train the model is provided. They are inspired by the structure and the behaviour of biological neural networks in the sense that the basic unit called a neuron or a node are connected to neurons of adjacent layers through modifiable weighted connections, resembling in some way synaptic connections in the brain. Neurons in the same layer are nothing more than components of a vector acting in parallel, being the layer the vector itself.

The role of a neuron in an artificial neural network is to send and receive inputs from other neurons or from an external source and compute an output that will serve as input to other neurons or produce a final outcome. Additionally, before delivering the output of a neuron to the next layer of neurons, a non-linear function also known as activation function is applied to the intermediate output to introduce non-linearities in the result. This is very helpful because most models that can be made of real world phenomena are non-linear.

Artificial neural networks are able to recognize patterns from the data provided and based on that be able to make predictions with high accuracy. To achieve this it is said that the

network must be trained. Basically, training a neural network consists in minimizing the value obtained from a loss function, also called the "error". In supervised learning, this cost function evaluates, given the input data, how well the network makes the prediction of the expected output. Some methods exist to minimize this error, relying in the determination of the gradient of the cost function and the update of the values of the weights of the connections between neurons. The most common method for updating the weights in a artificial neural network is called backpropagation. Successive adjustments of this weights will cause eventually the network to produce accurate predictions, even when the data fed to the network come from a different source than the data used during the training phase. After meeting certain criteria, which usually has to do with surpassing some evaluation metric value (accuracy or precision for example) or running the algorithm for a prefixed number of epochs (a cycle of training using all the training data), the training can be stopped.

2.1.3 Deep Feedforward Networks

Feedforward networks are models whose goal consists in approximating a function $f*$ which maps an input $\mathbf{x}$ to a specific output y, provided in supervised learning tasks [22]. To map this function, feed forward networks must learn the value of some parameters $\boldsymbol{\theta}$ that yield the best approximation:

$$y = f(\boldsymbol{x}; \boldsymbol{\theta}) \tag{2.1}$$

The reason these models are called feedforward networks has to do with the direction in which the calculations are done, which is forward, starting in the input layer where the function that is being approximated is evaluated with all the examples belonging to $\mathbf{x}$ one at a time, then in the hidden layers where the actual function is being approximated and finally in the output layer, where the prediction is done, i.e. where a value that ideally should be close to y is obtained. The most simple feedforward neural network is the Multilayer perceptron, conformed of one input layer, one or more hidden layers and an output layer.

Prior to training a feedforward network on a dataset, it is essential to define the following aspects. First of all, the architecture of the network must be defined. This step includes deciding the number of layers and how many neurons will be contained in each these layers. Another important aspect to be defined is the way these neurons will be connected between layers, basically hidden layers could be fully connected or partially connected. When the

neurons of a layer receive an input from all the neurons of a previous layer, the layer is also called dense layer.

Other aspects needed to be taken into account are the appropriate choice of the activation functions that will serve to compute the output values of each hidden layer's neurons, the loss function used to calculate the loss or the error after a prediction has been made and the optimizer which serves to minimize the error obtained.

2.1.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a subtype of feedforward neural network specialized in working with data such as images and time-series data like audio signals, both which can be thought as tensors. For example, grey scaled images are basically two-order tensors, i.e. matrices, while RGB images can be represented as three-order tensors since a RGB image is composed of three grids, one for each color. Audio signals, on the other hand, can be seen as first-order tensors, i.e. vectors.

The main feature of convolutional neural networks is that the data fed into the network passes through a series of layers called "convolutional layers". Convolution is a linear operation between two tensors of any rank, for neural networks, an input which could be an image or an audio signal and a kernel (also called filter), usually much smaller than the input. For a second-order tensor, e.g. an image, and a two-dimensional kernel the discrete convolution would be written in the following form:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.2}$$

being the first term the input (the image) and the second term the kernel. The output of this operation is referred as the feature map, an intermediate representations of the input which is also a tensor. The indexes m and n correspond to the dimensions of the kernel while i and j the indexes of the pixels of the image. In the case of having a three-order tensor as input, an extra dimension should be spanned using a three-order tensor as kernel. Furthermore, convolution operation satisfies the commutivity property:

$$s(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.3}$$

In this case the kernel has been flipped relative to the image, however this equation can be implemented more easily because there is less variation in the range of valid values of m and n. There is a similar operation called cross-correlation that usually is used indistinctly as convolution but avoids the inconvenience of having to slide a flipped kernel across an image:

$$s(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.4}$$

The principal difference between dense layers and convolutional layers is that dense layers perform matrix multiplications to compute the interaction between each input and the output neurons, while convolutional layers improve efficiency by reducing the number of operations and memory usage by a concept known as sparse connectivity. The way this is done has to do with was mentioned above about using kernels much smaller than the inputs. Actually, from using small kernels it is possible to extract meaningful patterns from a much bigger input. The size of the filter determines the receptive field, that is the number of input neurons that affect an output neuron. By using fewer parameters, sparse connectivity also allows to reduce the amount of memory needed.

CNNs are especially successful in tasks related to image classification. A CNN learns to detect simple patterns such as edges in its first layers, increasing the complexity of such patterns in subsequent layers. This is achieved going through the image (input feature map), applying the convolution (or the cross-correlation) operation, which is the sum of the terms obtained from the element-wise multiplication between the kernel and the corresponding local window, i.e. the chunk of the image covered by the kernel. From this operation an output image of smaller dimensions is produced. This output focuses on the regions that exhibited the characteristic that was being sought to highlight through the filter. Furthermore, it serves as input for the next layer, for which the edges detected in the previous layer are used to detect slightly more complex forms; thus the complexity of the features will increase as the input goes forward through subsequent layers. This process can lead to learning patterns as complex as shapes which resemble a human face.

Usually after a convolutional layer an activation functions is applied to the output for the same reason that it is applied after a dense layer, to add non-linearities to the network. A common activation function is the rectified linear activation function, also known as ReLU. After applying a non-linear function a pooling layer comes next. The purpose of this layer

is to further reduce the number of parameters and the number of operations through the generation of a compact statistic representation of the input. Methods such as the max function or calculating the average are calculated using kernels within fixed-size regions of the input. Basically, pooling is a downsampling operation.

An important concept used both in convolutional and pooling layer is the concept of stride. Stride is a parameter that modifies the position of the kernel over an image by sliding it a fixed number of pixels along all the available axis. For example, Figure 2.1 depicts the process taking place in a pooling layer using a max function for a 4x4 matrix using a 2x2 kernel with stride (1,1).

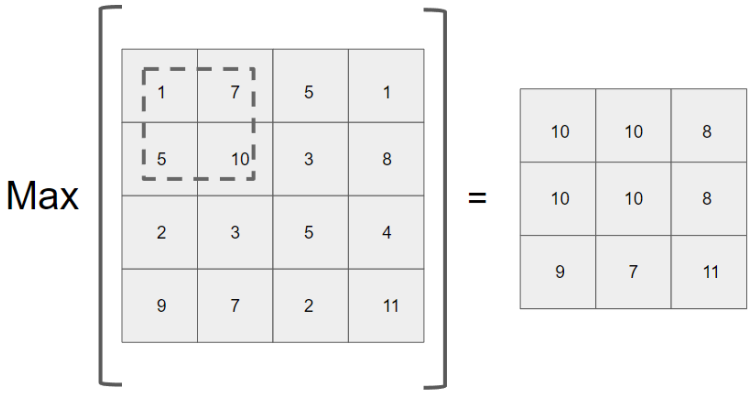


Figure 2.1: Max function being applied to a 4x4 matrix using a 2x2 kernel and with stride (1,1).

The arrange of this layers depends on many aspects, mainly the type of task, the type and the size of the inputs and even the expertise of the programmer. After some convolutional and pooling layers, a flatten layer is placed to squish the feature map information into a one dimensional vector. This vector feeds the last layers which normally are fully connected layers. Finally, a loss function is applied for backward error propagation.

2.1.5 Long Short-term Memory Netoworks

There are other types of neural networks besides multilayer perceptrons and convolutional neural networks. Long Short-term Memory Networks or LSTM networks are a different type of neural network that belong to a family of models known as Recurrent Neural Networks (RRNs). RRNs are capable of learn from sequential data like time series or text, through

feedback connections which act as a memory. This means that the current output is influenced not just by the current input but also by information seen in prior time steps.

RRNs are similar to common Neural Networks. They include an input layer, hidden layers in the middle and an output layer. The difference has to do with RNNs neurons having the ability to use a new feature called hidden states, which are sort of photographs of the previous input data. Classical RNNs use the hidden state from the previous time step, combining it with the current input and passing the result to an activation function in order to obtain a new hidden state to be used in the next time step. The inconvenience with this type of RNN is that as more time steps, older information vanishes with newer hidden states. Also, classical RNNs are prone to exhibit unstable behaviours due to a problem known as vanishing gradient, which has to do with the output of certain derivatives involved in the training process reaching values close to zero.

LSTMs are useful for solving this problem. They introduce new components to its architecture, mainly a memory cell comprised of weights and gates (multiplicative units). These gates include a forget gate, which decides what information can be discarded from the cell state, an input gate, whose function is selecting some values from the input to update the cell state and an output gate, which filters part of the information from the cell output. To fulfil their respective tasks, these gates use non-linear functions, specifically, sigmoid and tanh; the first to keep or discard certain information and the second to avoid vanishing gradients. This allows LSTMs to remember information beyond the previous time step, which translates in remembering longer sequences of data; in fact, the cell state gathers information from all the previous time steps.

2.2 Sound Classification

Neural networks are powerful tools that are being used to solve different problems, such as classification tasks, which given certain conditions may be considered as supervised learning tasks. Classification tasks consist in identifying the correct category of a set of observations based on a dataset whose role is to function as training data. Data classification allows solving specific problems by using specific types of data, e.g., sound.

2.2.1 Environmental Sound Classification

Environmental sound classification is a known problem that belongs to the audio recognition field that requires the implementation of various methods related to the fields of audio signal processing and machine learning. To be clear, the word classification implies that for a single input the output of the algorithm can only be one of the possible classes that make up a dataset.

The purpose of this task is to categorize small audio clips or recordings in one of the available classes of a dataset. Typical systems extract acoustic features from the pre-processed audio signals to then perform the proper classification using supervised classifiers such as neural networks. Unlike other sound recognition tasks such as Music Information Retrieval or Automatic Speech Recognition, both dealing with sounds which have a proper structure and a considerable Signal to Noise Ratio (SNR), a measure which can be defined as the ratio of the signal of interest to the level of background noise, environmental sound recognition deals with unstructured sounds that in addition possess a small SNR, making the classification task more complex.

Different approaches have been studied in order to implement accurate systems for solving this task, including the utilization of diverse machine learning models such as Support Vector Machines, K-Nearest Neighbours and Random Forests. Deep Learning models have also been employed for this task giving even better results. An important thing to bear in mind when using neural networks for single-label classification tasks is that classes should be mutually exclusive so the outputs of the neural network be interrelated. This condition can be achieved using a softmax activation function in the output layer which normalizes the output for each class to sum up to one.

2.2.2 Environmental Audio Tagging

Environmental Audio Classification can be expanded to output more than one class for a single audio. This task is also known as Audio Tagging and it is a multi-label classification problem. Datasets for this task are made up by recordings in which multiple classes of sounds are present in a single frame.

An environmental sound classification neural network can be adapted to Audio Tagging with minor modifications in its architecture. The most important change is the use of

a sigmoid activation function (instead of softmax) in the output layer and binary cross-entropy (instead of categorical cross-entropy) as loss function. The sigmoid function allows to calculate independent probabilities for each of the possible classes in the range from 0 to 1. The presence of a class in a single audio is determined by a threshold normally fixed at 0.5. If the probability obtained for a class lies below this value, the class is considered absent in the audio. On the other hand, a value grater than 0.5 would mean that the class of sound is present. The value of this threshold could be augmented or decremented if some bias is detected towards false-positives or false-negatives, respectively.

2.3 Feature Extraction

It is important to keep in mind that the input to a neural network usually is not going to be the raw audio signal, although it could. Rather, what feeds the network is a compact representation of the signal obtained via some mathematical procedure. This compact representations of sound are also called acoustic features and their advantage over raw signals is that they take less memory space and need less computational power, incrementing processing capability. There are various types of acoustic features but in general all need to fulfil two properties in order to make the learning problem much easier which ultimately will lead in a good performance of the classifier, the first one being low variability between features obtained from samples of the same class, second, high variability between features obtained from samples belonging to different classes.

Acoustic features discard meaningless information of a signal, keeping just the information which best reflect its physical properties. Examples of features are temporal features, computed directly on the temporal waveform, spectral features which rely on the frequency content of the signals, and cepstral features which are the result of applying the inverse Fourier transform of the logarithm of the signal spectrum.

2.3.1 Sound

Sound is a type of energy which propagates through an elastic medium that could be any type of gas, liquid or even solid. It's produced when a force makes an object vibrate. These vibrations induces adjacent particles to move back and forth in the same direction in which energy is being transported, i.e. the same direction that the sound is travelling. This creates high and low densities areas within the medium called compressions and rarefactions, also

referred as pressure waves. Humans detect sound because at some point these vibrations reach the eardrums and make them vibrate too. This vibrational energy is then transformed into electrical impulses that end up reaching the brain through the auditory nerve.

Furthermore, Sound carries a great amount of information about the environment where they occur at the instant in which they are taking place. This information may encompass individual physical events but also complex events which may represent a whole scene consisting of multiple sounds in the foreground plus background noise. The overlapping of sounds can make the task of differentiate each element difficult for humans and machines.

2.3.2 Digital Sound

Sound could be detected as analog signals or digital signals. Unlike humans, who process sound naturally in its continuous form, computers implement mathematical models to make discrete representations of sound. This discretization of sound is also called "Digitalization". Digitalization translates analogue signals from the physical world into sequences of numbers which computers are able to process. Converting an analog signal into a digital signal (A/D conversion) requires the following three steps:

1. Sampling the signal at uniform time steps, i.e. at a uniform frequency [20]. According to the Nyquist criterion, the lowest sampling rate permitted in order to reconstruct an analog signal with high fidelity should be more than twice the highest frequency component:

$$f_n = 2f_{max} \tag{2.5}$$

   When the sampling rate is too low, audio signals carry the risk of loosing some information or being distorted, phenomena known as aliasing.

2. Quantize the acquired samples. First, a signal range must be divided into a fixed number of intervals of the same size. The number of intervals is determined by the bit resolution also known as bit depth. Classical bits, which are the most basic unit of information in computing, can only take two values, zero and one. This means that if the bit depth is n, a signal range is divided into 2^n intervals. Then, a quantized value is assigned to each interval and every sample is rounded up or down to the nearest

one. Obviously, the smaller the number of bits the greater the numeric distance some of the sample values will need to be rounded. This distance is called quantizing error.

3. Coding the values obtained during the quantization to binary code or other machine language. This values are inputs that a computer can accept.

Converting an analog signal into a digital one involves quantizing both axes, by means of the sampling rate and digitalizing levels (Figure 2.2).

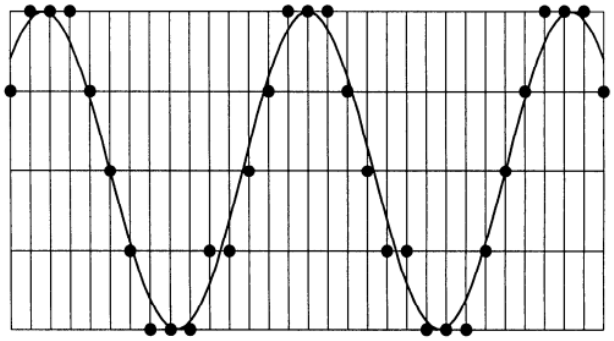


Figure 2.2: Example of the conversion of an analog signal into a digital one using five digitizing levels and acquiring 14 samples per cycle. Image taken from [23].

2.3.3 Fourier Transform

Once a sound wave is converted into a digital representation, i.e. an audio signal, many interesting things can be done in order to extract valuable information from it. By application of a mathematical tool called the Fourier transform (eq. 2.6), any non-periodic wave, e.g. an environmental sound, can be decomposed in its constituent periodic waveforms, which are basic sine waves of different frequency values. In practice, an approximation of the original signal can be obtained by a linear combination of a finite number of constituent waves. As more terms are added, the sum gets closer to the original wave.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \mathrm{e}^{-j\omega t} dt \tag{2.6}$$

In order to obtain the constituent sine waves, the Fourier transform maps a function in the time domain to a complex-valued function in the frequency domain. To observe which frequencies are present in a signal it is necessary to calculate the amplitude of the sine wave

for the frequency of interest by calculating the magnitude or absolute value of the complex value. If a value of zero is obtained, it means that the frequency is absent. Furthermore, it is possible to construct a diagram displaying the amplitudes of all the sinusoids for all the frequencies, this is also known as the spectrum of the waveform.

The Fourier transform is extremely useful for working with continuous non-periodic waves, however, as mentioned in the previous section, computers don't process continuous signals, since microphones essentially measure air pressure at many different points in time. There is another transformation called the Discrete Fourier Transform (equation 2.7) used when the available information of a signal is a set of values acquired at instants separated by a fixed time interval (sampling). However, in practice it is preferable to implement the Fast Fourier Transform which is an optimized implementation of the Discrete Fourier Transform.

$$F(\omega) = \sum -\infty^{\infty} f[n] \mathrm{e}^{-j2\pi\omega n} \tag{2.7}$$

2.3.4 Short-time Fourier Transform

Furthermore, the Fourier transform can also be computed over short overlapping time windows, preferable in situations where the frequency components of the signal vary over time; this is transformation is also known as the short-time Fourier transform (STFT) and it can be written in the following form:

$$F(t, \omega) = \sum 0^{N-1} h[k] f(tN + k] \mathrm{e}^{\frac{-j2\pi k\omega}{N}} \tag{2.8}$$

where h[k] is a window function that is zero-valued outside a chosen interval. This type of functions can take different shapes, being one of the most common the rectangular and the Hamming windows. The dynamic of the STFT consists in calculating the Fourier Transform over a window and then move it some distance, preferably smaller than the frame length, introducing a certain degree of overlap that will serve to obtain a smoother representation of the transformation with some statistical dependency between adjacent frames.

2.3.5 Spectrograms

In addition to what was explained in the previous section, the magnitude of the frequency spectrum of the signal for each frame can be obtained and plotted as a 3D graph, with time,

frequency and magnitude as the components, but with the subtlety that the magnitude is represented by colors instead of another spatial dimension. This kind of representation is known as a spectrogram and is just another way of representing the output of applying the short-time Fourier transforms through an audio signal.

In simpler terms, spectrograms are visual representations of sound. A spectrogram shows which frequencies make up a sound signal and the way they vary over time. A collection of spectrograms can be used as input to train a machine learning model and perform a classification task, however, from this representation of sound more powerful features could be extracted.

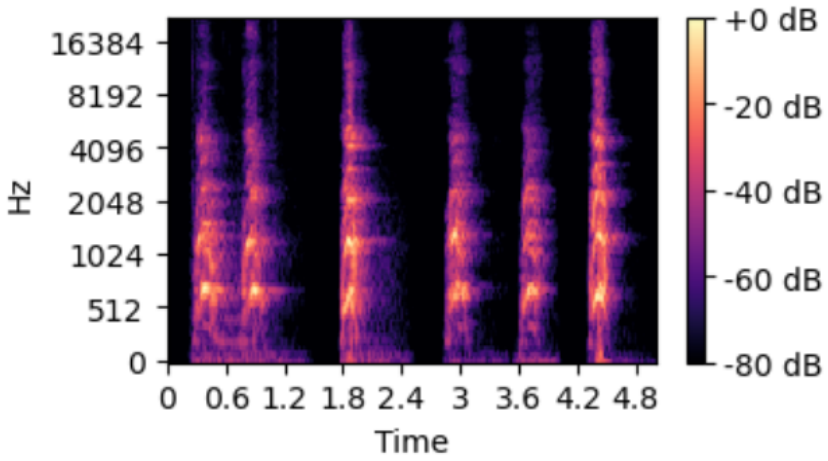


Figure 2.3: Spectrogram image of the barking of a dog. Six barks can be easily identified from the image.

One of most common acoustic features is the mel-spectrogram (Figure 2.3). These features are obtained with the purpose of representing the spectral content of an audio signal, just like normal spectrograms. The difference has to do with the frequency values being mapped from a linear frequency scale to a scale known as the mel scale, designed to represent the way humans perceive sound, which is logarithmic in nature. Studies have shown that humans are better discerning low frequencies than high frequencies separated the same distance. One example of a relation between Hertz and mels is represented by the following equation:

$$mel(\omega) = \frac{1000}{log2} \log(1 + \frac{\omega}{1000}) \tag{2.9}$$

The process of converting a spectrogram into a Mel-spectrogram is not as simple as just using this equation for each individual frame of the spectrogram. First, a set of overlapping triangular mel-scaled filters (a filterbank) is applied to the spectrogram in order to produce frequency bands. The size of this filters is determined by the mel scale that is being used, but in general these filters are narrower at low frequencies than at high frequencies. This has the consequence that low frequencies are emphasized over high frequencies.

The process can stop here if the desired features are the mel-spectrograms. However, there is an additional step that can be computed in order to obtain more compact features known as Mel-frequency Cepstral Coefficients (MFCCs). To obtain MFFCs it is necessary to decorrelate the filterbank coefficients by applying the inverse discrete cosine transform of the log energy of each of the mel frequency bands. Typically only 13 coefficients are kept while the rest are discarded, however, this is not mandatory, in some cases more coefficients can yield better results.

In practice, classification tasks relying on algorithms such as deep neural networks, have shown that MFCCs may not be the best features despise its success in tasks related to speech recognition. Compared to MFCCs, Mel spectrograms are constructed by a larger number of coefficients, which translates in a higher frequency resolution, often needed for tasks of this complexity.

2.3.6 Datasets

Datasets are of most importance for any supervised learning system. They are decisive for reaching high accuracy or other evaluation metric values. Audio datasets used for supervised learning tasks are conformed of both the audio files and the metadata, normally a tabular file manually created where each column will represent an attribute of the audio files, for example, name, class, duration, fold, etc. Building a dataset for environmental sound classification is not as simple as it would seem. First, it has to be decided which classes will be contained in the dataset, a consideration that will depend on the subject of the research but also on some practical aspects, being the most important the resources that need to be invested during the data acquisition stage, which can be can be both time-consuming and expensive. Taking this into consideration should influence largely the decision on both the number of classes and samples per class to be acquired. Also, when working with environmental sounds, two scenarios are possible during the data acquisition stage; first, recording

the sounds of each class under controlled conditions, isolated from any background noise. The other scenario would be to collect the audio data in natural conditions, i.e. without avoiding other types of sound been captured also. Both types of datasets exists, being an example of the first type the ESC-10 dataset and UrbanSound8K an example of the second [24, 26].

Datasets have a decisively role in the performance of a classifier. In order to train successfully a supervised learning system for environmental sound classification, it is required sufficient amount of data of each of the classes. Also, the quality of the data is also an important aspect. Different sound sources usually exhibit variations in sound producing mechanisms which translate in differences in the produced sounds. Data with low variability will cause the system to present a behaviour known as overfitting. This means that the system will often fail in generalize to new-unseen data. However, having plenty amounts of data with high intra-class variability will facilitate the mapping between the acoustic features and the class labels, allowing to estimate the correct outputs when testing the system with new-testing examples.

In environmental sound classification is intuitive to expect that samples of the same class will show high variability, however it may not be enough. Variability of a sound source may depend in factors such as the characteristics of the acoustic environment, the distance between the microphone and the sound source, the capture device and the presence of interfering noises.

2.3.7 Evaluation Metrics

Good performance in a classification task is measured by how often the model used classifies data correctly. The most common evaluation metric used is accuracy, defined as the percentage of correct predictions made by the model from the total number of observations (data samples). However, accuracy alone doesn't give sufficient information to conclude if a model has learned sufficiently well how to perform on the task it was programmed for.

Furthermore, in a multi-label classification task, it can be argued that accuracy may not be a useful metric. Take the example of an audio tagging system which tags correctly all existent sounds present in a recording but also indicates as present an absent sound. Normal accuracy will categorize this prediction as incorrect although the system performance was not all that bad.

An alternative is to use a different metric. One that is more appropriate for this type of tasks is the Hamming Loss, defined as the proportion of incorrectly predicted labels to the total number of possible labels. In other words, Hamming loss penalizes individual labels. The way it does this is by using the logical operation XOR between the real and the predicted labels for each instance and then calculating the average across the entire dataset (eq. 2.10).

$$HL = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} X_{i,l} \bigoplus Y_{i,l} \tag{2.10}$$

where $\bigoplus$ is the XOR operator, N the number of instances of a dataset, L the number of classes, $Y_{i,l}$ the i-th prediction for the l-th class and $X_{i,l}$ is the l-th real label for the i-th instance.

There are other metrics that could help in getting a full picture of the behaviour of the model being tested. This metrics are the precision, recall and the f1-score, the latter which depends on the first two. Precision and recall are simply the percentage of correct positive predictions over the total positive predictions made and the percentage of correct positive predictions over all actual positive observations, respectively. The f1-score is just the harmonic mean of the two. For the binary case, i.e. single-label classification, computing this metrics (including accuracy) is pretty straightforward using the concepts of True Positive (TP), i.e. correctly predicting the positive class, True Negative (TN), i.e. correctly predicting the negative class, False Positive (FP), i.e. incorrectly predicting the positive class, False Negative (FN), i.e. incorrectly predicting the negative class. The formulas are shown below:

$$\text{Accuracy} = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{2.11}$$

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{2.12}$$

$$\text{Recall} = \frac{TP}{(TP + FN)} \tag{2.13}$$

$$\text{F1score} = \frac{2(\text{Precision})(\text{Recall})}{\text{Precision} + \text{Recall}} \tag{2.14}$$

For multi-label classification tasks each of this metrics have to be computed for each class

and then use some aggregate metrics like macro, micro, weighted and sampled averages to obtain a good insight of how the model is behaving.

2.3.8 Data Augmentation

Training deep learning models most of the time requires large amounts of data to encompass the high variability that certain types of data exhibit. In some cases, datasets that could be considered large don't possess sufficient instances to train a model sufficiently well to give accurate predictions. Environmental sounds classification is a task that suffers from this drawback. In practice, the same sound can be recorded varying many parameters, for example, the distance from the source to the acquisition device, the acquisition device itsel and the presence or absence of background noise. In general, all this possible variations increase the variability that a sound may exhibit, which makes high reliable systems difficult to achieve.

To address the inconvenience of having insufficient instances of data, data augmentation techniques based on the application of signal transformations can be used to generate new data samples. Some basic transformations include pitch shifting, random noise addition and time stretching. A more sophisticated method has to do with the utilization of deep generative algorithms. Generative Adversarial Networks are an example of this type of algorithms.

2.3.9 Generative Adversarial Networks

Generative adversarial networks, also known in English as GANs, are a class of neural network architecture used to synthesize data. Announced in 2014 by Ian J. Goodfellow, nowadays GANs are been used to synthesize diverse kinds of data, from realistic images of human faces or works of art to digital audio such as music, environmental sounds or spoken languages. GANs have also been used to synthesize molecules with specific pharmacological properties. It is a technology that is becoming extremely popular even though it is still in early stages of development.

A generative adversarial network is made of two main components: a generator model and a discriminator model, both being normally neural networks. The role of the generator is to capture the probability distribution of the training data in order to be able to generate new instances. The discriminator, on the other hand, must determine if the samples shown to it

come from the original dataset or were produced by the generator. Both models are trained simultaneously in an "adversarial" way which basically means that to train correctly the generator, the probability that the discriminator commits a mistake should be maximized while at the same time the probability that the discriminator correctly discerns between real and fake (produced) data is also maximized. The dynamics of this competition encourages both models to improve their performance progressively.

As stated, a generative adversarial network can learn to replicate the distribution of probability of a set of data and synthesize new instances of it. The process consists in generating a vector $\mathbf{z}$ of random values obtained from another known distribution, for example, a normal distribution or a uniform $p_z(\mathbf{z})$ distribution. This latent vector is then mapped to the distribution of the dataset by feeding the generator $G(\theta_g, \mathbf{z})$ with it. The use of a neural network has the advantage that it is not necessary to clearly know the shape of the data distribution. To train the generator, a discriminator model $D(\theta_d, \mathbf{x})$ is also implemented. The output of this network $D(\mathbf{x})$ represents the probability that $\mathbf{x}$ is a sample synthesized by the generator.

Regarding the training dynamic of the original generative adversarial network proposed by Goodfellow, the generator must minimize the expression $log(1–D(G(\mathbf{z})))$ by the following equation:

$$min_G max_D \, V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))] \tag{2.15}$$

Ideally, the network stops reaching the point where the generated data becomes indistinguishable from the original data. However, the problem with classic GANs is that both the generator and the discriminator suffer from instability during the training. Both networks update their loss functions simultaneously and independently and this has the consequence that there is no guarantee of convergence. In addition, system performance is very sensitive to the choice of hyperparameters, which can easily lead to system destabilization.

Controlling discriminator performance using this approach is particularly difficult. Originally, the Jensen-Shannon divergence was used as the loss function of the discriminator to determine the distance between the distribution of the generator and the target distribution. However, the Jensen-Shannon divergence is not always continuous with respect to the parameters of the generator, leading to the phenomenon known as vanishing gradient.

To remedy some of these deficiencies, the behaviour of generative adversarial networks had been analysed using different loss functions. The Wasserstein loss function, based on a metric known as the Wasserstein distance can be used to prevent vanishing gradients and give more stability to the two models during the training.

2.3.10 Wasserstein GAN

To improve the performance of the generative adversarial network, it was proposed to change the loss function by one that had the property of being continuous and differentiable, providing a linear gradient even when the discriminator was well trained. A loss function that accomplishes this is the Wasserstein loss. Proposed in 2017, the Wasserstein loss was an alternative to the Jensen Shannon loss. It is based on the metric known as the Wasserstein distance. This metric calculates the distance between two probability distributions in terms of the cost of converting a distribution in the other.

The benefit of the Wasserstein loss over other loss functions, for example, cross entropy loss, is that it provides useful gradients almost everywhere in its domain, allowing continuous training of the models. However, to ensure that this measure is valid, it is required to guarantee that the discriminator complies with the property of continuity 1-Lipschitz. To achieve this, constraining the discriminator weights was suggested. However, in the same work it is concluded that this strategy is not sophisticated enough to guarantee the model convergence. As an alternative, it was proposed to replace the constriction method of weights with a gradient penalty that enforces the same condition. This allows a more stable training and requires very little tuning of the network hyperparameters.

It is important to mention that the implementation of the Wasserstein loss function changes the notion of the discriminator to that of a critic, since now the output of the model is a real number that represents the quality of the generated sample instead of a probability that the sample would have been synthesized by the generator.

2.3.11 WaveGAN

WaveGAN is the first model of a generative adversarial network that was used to synthesize audio signals. Its architecture is based on the DCGAN model (Deep Convolutional Generative Adversarial Networks) that popularized the use of GANs to synthesize images. The main feature of the DCGAN has to do with the use of the transposed convolution operation,

which generates low-resolution feature maps from a low dimensional latent vector in order to build a high resolution image.

One of the main differences between DCGAN and WaveGAN has to do with the use of one-dimensional filters and the use of Wasserstein loss to ensure stability during training, instead of two-dimensional filters and the use of cross-entropy as loss function. WaveGAN uses filters of length $(25, 1)$ instead of squared filters of size $(5, 5)$. The stride, which is the distance the filter move each step, also grows in one of the two dimensions and is reduced in the other; it changes from size $(2, 2)$ to $(4, 1)$. These changes result in the WaveGAN having the same number of parameters and the same number of output values as a DCGAN ($64 \times 64 = 4096$). However, WaveGAN adds an additional transposed convolution layer to return samples of 16384 values, a little over a second long at an equivalent sample rate of 16 kHz.

CHAPTER 3

Methodology

In this section are described all the steps that were performed, the software and the equipment used to achieve the objectives proposed in this thesis. The sections are divided in a way that first all the processes that involve the preparation of the original audio files are explained, including all the data augmentation techniques that were explored. The following sections are dedicated to the main objective of this work which is the classification of environmental sounds, starting with the monophonic approach, a single-label classification task, and then the polyphonic approach, a multi-label classification task, including a complete description of how a dataset of polyphonic sounds was created from the original ESC-10 dataset.

3.1 Hardware

For this work, all of the programs were written and tested using an Asus TUF Gaming A15 obtained with the scholarship awarded by the master's program. The specifications of this PC are the following:

- CPU AMD Ryzen 5
- 24 GB of RAM memory
- GPU Nvidia GTX 1660 Ti

3.2 Software

All the code was written using Python language, to be precise, version 3.9.12. The principal libraries that were used were Numpy, for manipulating data, the high-level API of Tensorflow 2, also known as Keras, for being a free open source library for artificial intelligence, Librosa and SoundFile for reading, processing and writing all the audio files and Scikit-learn for calculating all the metrics for evaluating all the classification tests. Other libraries that were used are Scipy, Matplotlib, Tqdm and Pandas.

3.3 Dataset

ESC-10 dataset was chosen to be used in this work. It consists of 400 audio files from ten different classes of sounds divided in five folds. The characteristics of this dataset are the following [24]:

- Duration of five seconds for each file.
- Sampling frequency of 44.1 kHz.
- Single channel (mono).
- Ogg Vorbis compression format at 192 kbit/s.
- 40 audio files per class.
- 8 audio files per class, per folder.

The sounds included represent three types of sounds with different characteristics:

- Transient sounds: sneezing, dog barking and clock.
- Sounds with high harmonic content: crying babies and rooster.
- Quasi-structured sounds: rain, sea waves, fire, helicopter, electric chainsaw.

3.4 Monophonic Audio Classification

The main objective of this work is the classification of polyphonic sounds, that is, the correct identification of more than one class of sound present in a recording. However, since this was the first approach to the topic of Audio Classification, in order to analyse the complexity of the problem, it was decided to start with the classification of monophonic sounds, that is, recordings where only one class of sound is present.

3.4.1 Audio Preprocessing

Prior to performing the classification, each file of the dataset needs to undergo some manipulation. The preprocessing stage of the data includes the removal of "silences", the homogenization of the duration, a downsampling and finally, the extraction of features, process which is intended to discard redundant data and economize computing resources such as time and storage.

In the first step of the preprocessing stage all the silences were removed. Silences are considered regions where the signal amplitude lies below a certain threshold; these regions may also contain background noise which may affect the quality of the features that will be obtained later. To do the removal, the envelope of each signal was obtained and the threshold value was set so that every region below the chosen value was trimmed. For this work the threshold value was set to 0.0005.

Applying this process results in a version of the dataset where the duration of the audio signals is not homogeneous, so before exporting the clean signals the duration of each sample was fixed. First, a fixed value was set for final duration of the audio files. Depending on this value and the actual duration of the signal the program may trim or extend each of the signals. When a signal must be extended, a function called "loopAudio()" copies from the start of the signal to the instant in which, when pasted at the end, the desired length is reached. For the monophonic audio classification task the fixed length was chosen to be five seconds. After the length of the signals was fixed, a downsampling of each signal was performed, decreasing the sample rate from 44100 Hz to 22050 Hz. Finally, each sample was stored in its correspondent class folder.

3.4.2 Data Augmentation

3.4.2.1 Classical augmentation methods

In this work two classical augmentation methods were implemented to increase the size of the monophonic dataset, time stretching and pitch shifting. Each of this methods were applied one time to each file of the ESC-10 dataset. Librosa's "pitch_shift()" and "time_stretch()" functions were used to perform both operations on each of the original files.

For the time stretching method two situations were able to occur depending on the result of "tossing a coin", that the file became 0.5 faster or 0.5 slower. A fixed duration of the final audio was set to five seconds, so if the new duration is greater than the fixed value, i.e., it goes slower, a section of the file is trimmed to match the desired length (Figure 3.1). On the other hand, if the new duration of the signal is less than the fixed length, i.e., it goes faster, the function "loopAudio()" used in the preprocessing stage is called. Applying this method for each file of the dataset generates 400 additional files.

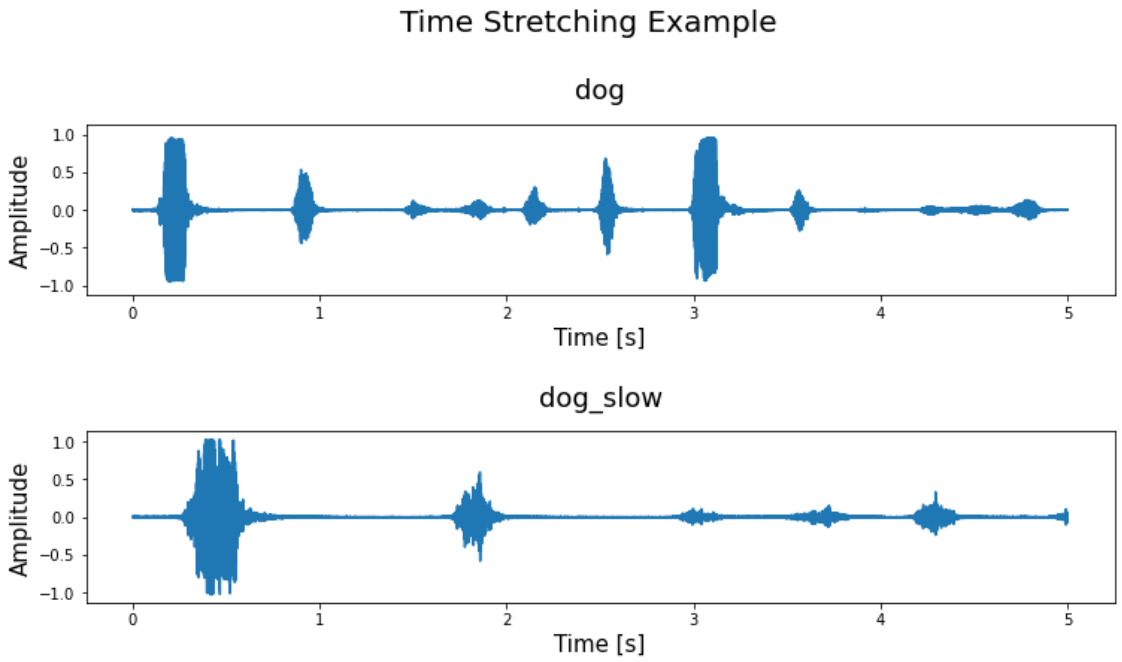


Figure 3.1: Example of the augmentation technique of time stretching. In the original recording (upper graph) the barking of a dog is unaltered while in the second image the barking has been slowed down.

The second method implemented was pitch shifting. In pitch shifting, the fundamental frequencies of a sound or an audio signal are increased or lowered by some unit interval, maintaining its duration constant. In the same way as with time stretching, for pitch shifting one new audio file was generated for each file of the ESC-10 dataset, creating another 400 files. To introduce some randomness, the number of semitones by which the entire signal is shifted up or down as a whole is defined also by a random number, with a possible range of values from -2 to 2, leaving aside the zero value since the resultant file would not suffer any change. An example is presented in Figure 3.2.

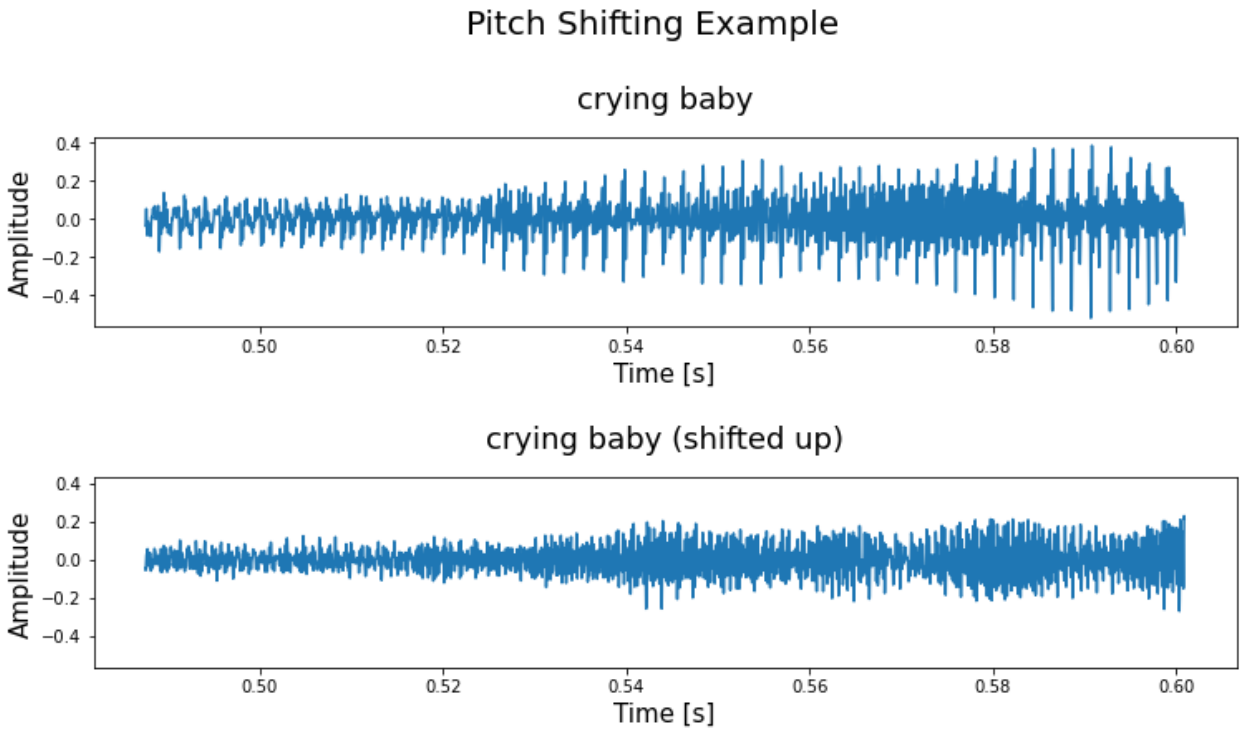


Figure 3.2: Example of the augmentation technique of pitch shifting. In the original recording (upper graph) the pitch of the cry of a baby is unaltered while in the second image (lower image) the pitch of the cry has been shifted up.

3.4.2.2 Wasserstein GAN + Gradient Penalty

A generative adversarial network was implemented as third data augmentation method. This GAN was programmed from scratch based on the WaveGAN model [18], which is a WGAN-GP (Wasserstein generative adversarial network with gradient penalty).

To be able to train the GAN some considerations must be taken. First, it must be ensured

the availability of GPU resources in order to be able to train the model in an affordable time. Also, the WGAN-GP needs to be trained for each of the ten classes, one by one; the problem is that, given the number of files available in the ESC-10 dataset is low and that only one class is used for each training, it is very likely that the network will exhibit a behaviour of overfitting which ultimately will result in poor performance. To solve this inconvenience to some degree, each five seconds-length file of the 40 available per class was split into five one second-length samples, giving a total amount of 200 files available per class for training the network. Additionally, to reduce computational cost each of the samples were downsampled from 44100 Hz to 16384 Hz, as suggested in [18]. This value corresponds also to the input dimensions of the discriminator model. Following these steps one should get a Numpy array of the form (200, 16384, 1). It is worth mentioning again that the WaveGAN model is trained with raw audio files (one-dimensional), not spectrograms (two-dimensional). The audio files were manipulated using Librosa and Pydub libraries.

Once the samples were ready, the next step consisted in defining the structure of the WGAN-GP inside a class named WAVEGAN. The architecture of the two neural networks contained inside this class were based on [18], however, one of the differences with their model is that in this work Leaky ReLu activation functions were used instead of ReLu to preserve the negative attributes of the audio signals. The discriminator output activation function was kept linear. The diagram of both generator and discriminator are shown in Figure 3.3 and Figure 3.4. As for the values assigned to the parameters of the WGAN-GP, which are shown in Table 3.1, they were the same as those used in [18], with the exception of the number of epochs, which in this work was chosen to be 1000 or 2000, depending on the subjective quality of the sounds when the thousandth epoch was reached.

Regarding the dynamic of the WGAN-GP training, it consists of updating five times the discriminator weights by an update of the weights of the generator. One iteration for training the discriminator include the following steps:

1. From a latent vector a set of fake sounds is synthesized by the generator.
2. A batch of real sounds, i.e., taken from the dataset, is passed to the discriminator.
3. The set of fake sounds is passed to the discriminator.
4. The gradient penalty is calculated.
5. The Wasserstein loss of the discriminator is calculated.

Name	**Value**
Batch size	64
Epochs	1000
Gradient Penalty (λ)	10
D updates per G update	5
Optimizer	Adam: $\alpha = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.9$
Phase shuffling	2
Latent vector size	100

Table 3.1: Chosen values of the main parameters used for the training of the WGAN-GP.

6. The gradient of the loss function with respect to the weights of the network is calculated.

7. Update of the weights of the network.

For the generator something similar is followed:

1. A batch of fake sounds is generated.

2. The discriminator is fed with the batch of fake sounds.

3. The loss is calculated using the loss function of the generator (different from that of the discriminator).

4. The gradient of the loss function with respect to the weights of the network is calculated.

5. Update of the weights of the network.

The class methods of the class WAVEGAN which represent the steps enumerated above, are listed below and a flowchart of the complete process is represented in Figure 3.5.

- getNoise(): To generate the fake sounds a latent vector of size 100 is generated with this function using a normal distribution in the range (-1, 1).

- apply_phaseshuffle(): Deletes noise artifacts that occur when making use of transposed convolution layers. This function "disturbs" the neural network in a way that it randomizes the phase of each feature map before entering the next layer of the discriminator [18].

- runGenerator(): Calls the "getNoise()" function to generate a low-dimensional latent vector and then pass it to the generator model. Return the output of the generator which is a set of synthesized sounds.

- runDiscriminator(): Passes a batch of fake or real sounds to the discriminator model. Returns the output of the discriminator.

- wasserstein_disc(): Computes the Wasserstein loss of the discriminator as the difference between the average discriminator score on real sounds and the average score on fake sounds, plus the gradient penalty.

- gradient_penalty(): Computes the gradient penalty from the gradient of a randomly weighted average between a batch of real and generated sounds.

- wasserstein_generator(): Computes the Wasserstein loss of the generator as the negative of the average critic score on the fake sounds.

Finally, the training routine is executed. For each epoch the indices of the training set are shuffled and split into batches. With 200 images it is possible to generate three batches of 64 and one of size eight. Due to the marked difference in sizes, the fourth batch is discarded. When the program is running, every ten epochs five sound samples are created and stored in a folder to keep register of the evolution of the sounds. Also, every 50 epochs the loss of both the discriminator and generator, plus the gradient penalty, are plotted to keep an eye on how the behaviour of the system evolves. For each of the ten classes the running time of the training (1000 epochs) lasts about six and a half hours.

Once the training has been executed for each of the classes, 200 one-second length files of each class are generated using the generator model with the corresponding set of weights. Every time a new file is created, a function verifies that the new sound exhibits active sound regions in order to avoid adding empty content files to the augmented dataset, which is possible. This quality check is done by obtaining the envelope of the signal and counting the number of samples whose amplitude lay above a threshold value set to be 0.009. If the number of samples is less than a the tenth percent of the total, the file is discarded and

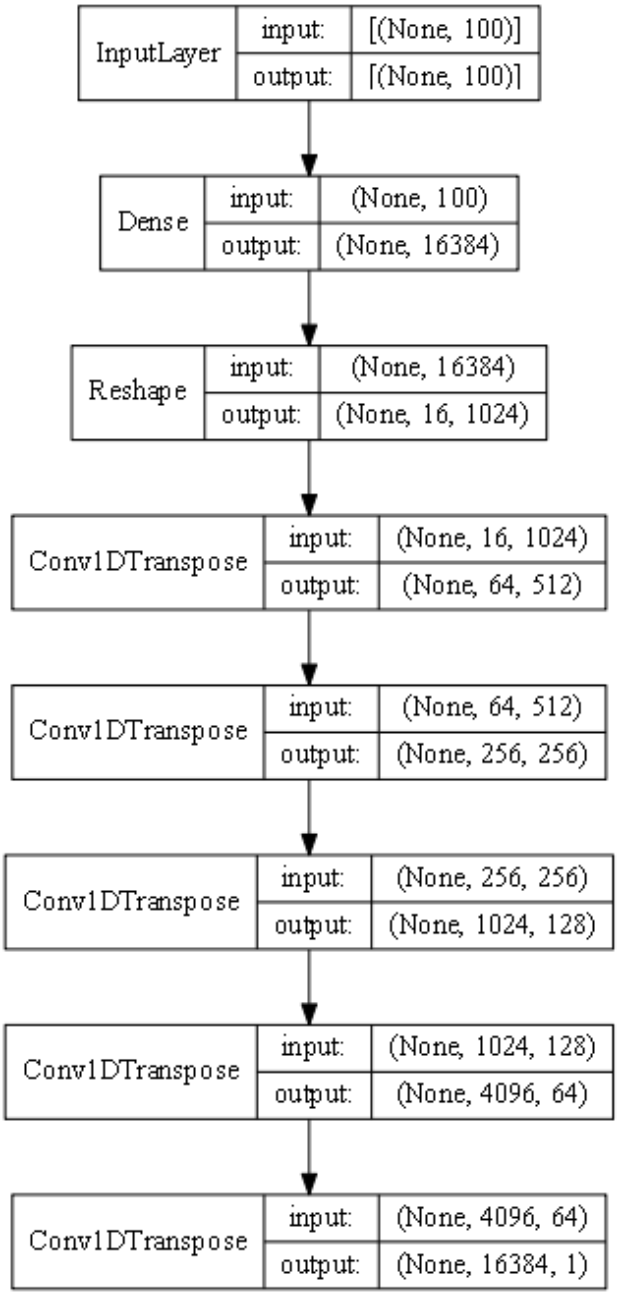


Figure 3.3: Generative model of the WGAN-GP. It consists mainly of 1D transposed convolutional layers using ReLu as activation function, with the exception of the last layer which uses a hyperbolic tangent function. Architecture based from [18].

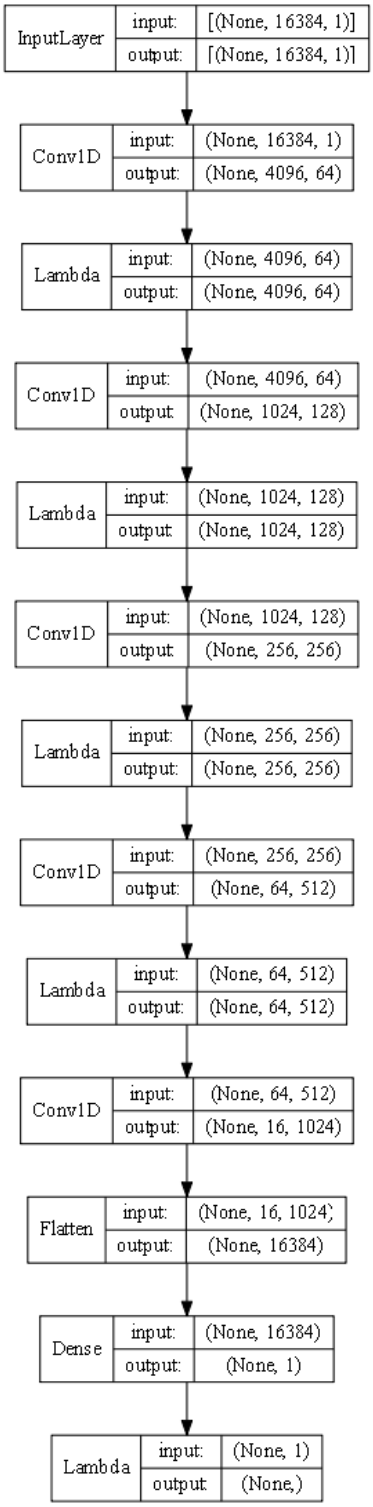


Figure 3.4: Architecture of the discriminator (or critic) model. It is build mainly upon 1D convolutional layers and lambda functions which apply the "phaseshuffle()" method to prevent the appearance of noise artifacts. Architecture based from [18].

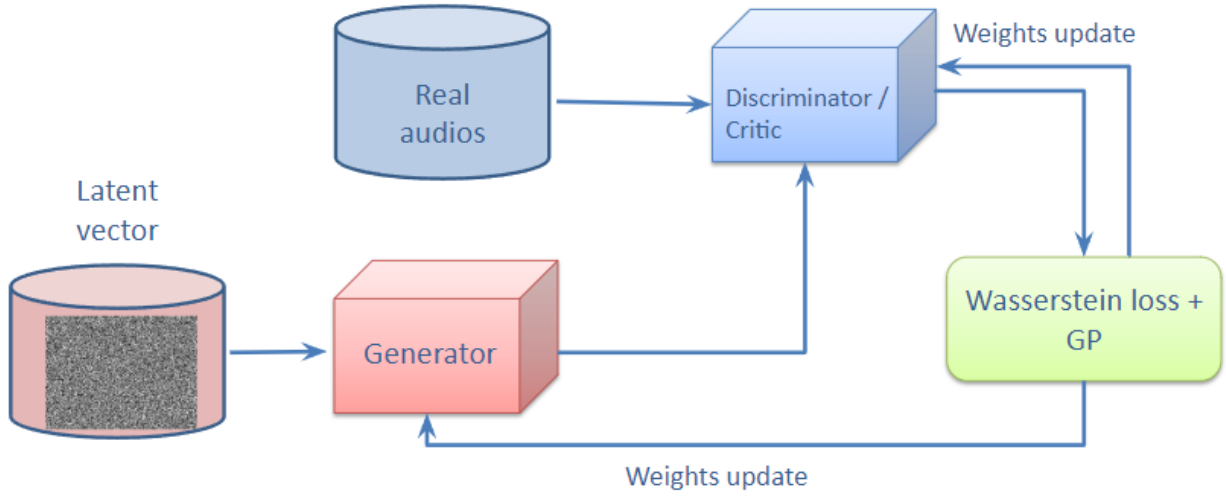


Figure 3.5: Flowchart of the dynamics of a Wasserstein GAN with gradient penalty. Image modified from [25]

replaced with a new one. After this process, 40 five-second length files per class are created by stitching together groups of five quality checked files, giving a total of 400 additional files to use in conjunction with the ESC-10 dataset.

3.4.3 Feature Extraction

Once the audio signals were pre-processed and the data augmentation methods applied, the next step was to extract the features, which as it have already been mentioned, are a compact and meaningful representation of audio signals. Two types of features were chosen as inputs for the classifiers, MFCCs and log-mel spectrograms. To obtain the MFFCs from the audio signals, "Librosa" library was used, specifically the implemented function called "mfcc" with a value of 26 for the number of filters that conform the filterbank, a value of 2048 for the number of samples included in each time frame, a hop size of 512 and a value of 40 for the number coefficients (MFFCs) that the function return. Those coefficients are then normalized using the Cepstral Mean and Variance Normalization method.

On the other hand, to compute the log-mel spectrograms, first, the mel-scaled spectrograms are obtained using a Hanning windows of size 2048 with a hop size of 512 and a value of 128 for the number of mel bands. For this, the function "melspectrogram" from the Librosa library was used. Then, to convert the scale of the amplitude axis of the spectrograms from a

linear scale to a logarithmic scale, the funcion called "power_to_db" is used to transform the power units (amplitude squared) into decibel units, which are logarithmic. An example per class of both types of features are shown in Figure 3.6 and Figure 3.7. In the next section, the classification of monophonic sounds will be addressed.

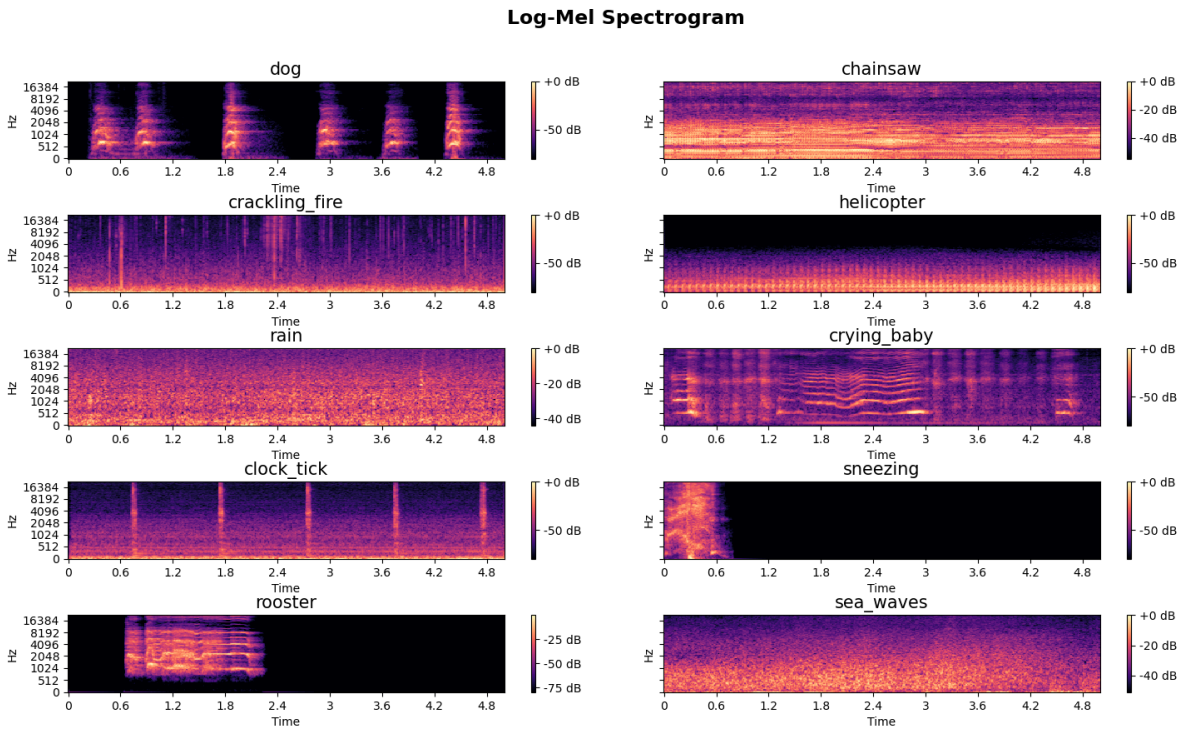


Figure 3.6: Log-mel spectrograms from each class of the ESC-10 dataset.

3.4.4 Classification

The classification of monophonic sounds was performed on two classifiers, a convolutional neural network (CNN) and a recurrent neural network, specifically, a Long Short-Term Memory (LSTM) network. The architecture of both is shown in Figure 3.8 and Figure 3.9. Both networks use categorical cross-entropy as loss function and Adam as optimization algorithm with a learning rate of 0.00001 for the CNN and 0.00002 for the LSTM network. Hyperbolic tangent activation function was used as activation function for the LSTM layer while convolutional and dense layers both implemented ReLu as activation function, with the exception of the output layer of each network which used a softmax function. The reason why softmax

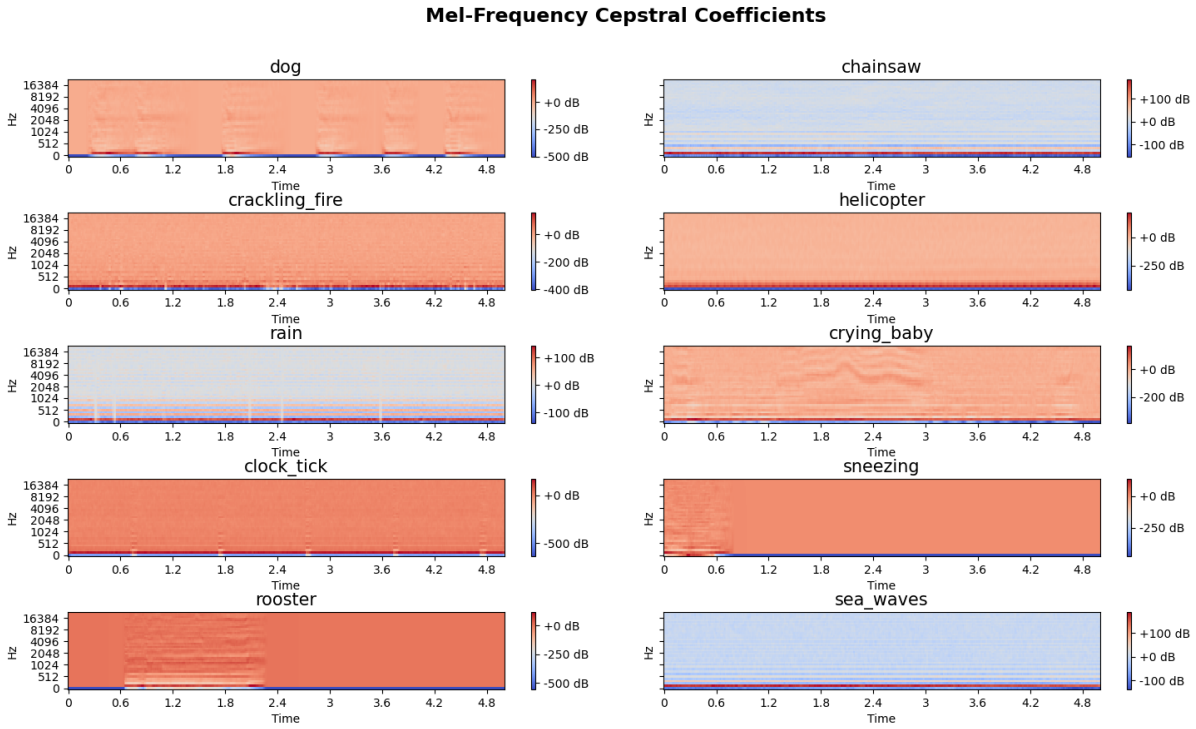


Figure 3.7: Mel-Frequency Cesptral Coefficients from each class from the ESC-10 dataset.

is assigned to the last layer has to do with the nature of environmental sound classification task, which is a multi-class classification problem where the classes are mutually exclusive, so one neuron is required in the output layer for each of the possible classes. By using softmax activation function in the last layer, the output vector is converted into a probability vector for which the sum of the probabilities is forced to yield a value of one. To be able to compare the predictions made by each model with the correct label of each sample of the testing set, all the categorical data was one-hot encoded, which means it was converted to binary vectors of zeros and ones.

As input, both neural networks received the features extracted from the cleaned audio signals. With two types of features available (MFCCs and Log-mel Spectrograms), two types of classifiers (CNN and LSTM) and four datasets available, which includes the original ESC-10 dataset and the three augmented datasets, all possible combinations were analysed in order to determine which yielded the best metrics. All the experiments done are shown in table 3.2.

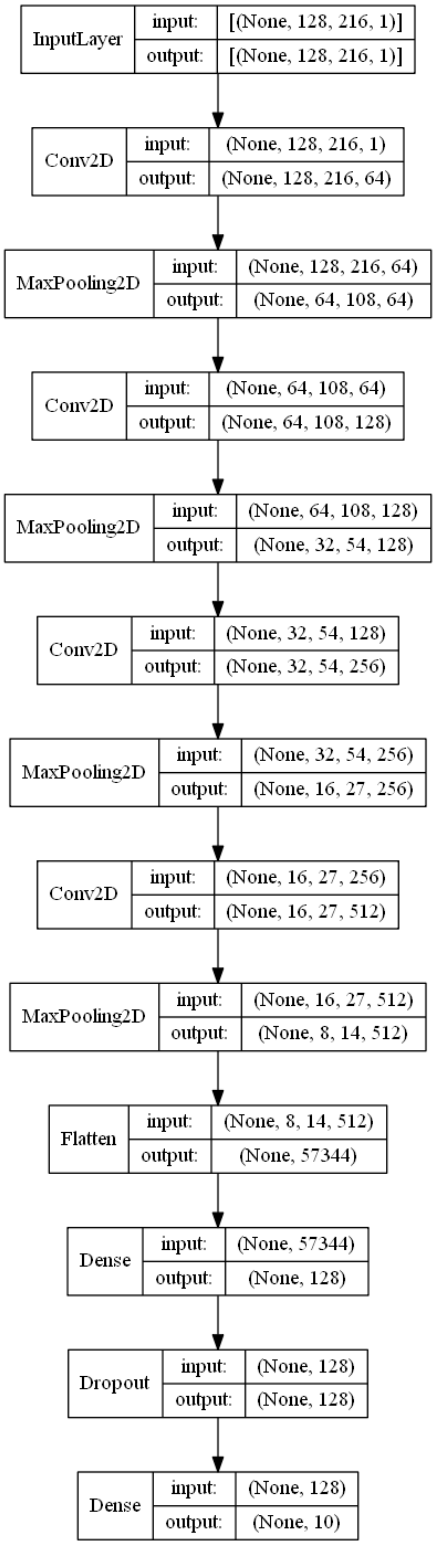


Figure 3.8: Proposed architecture of the CNN used for the monophonic classification task.

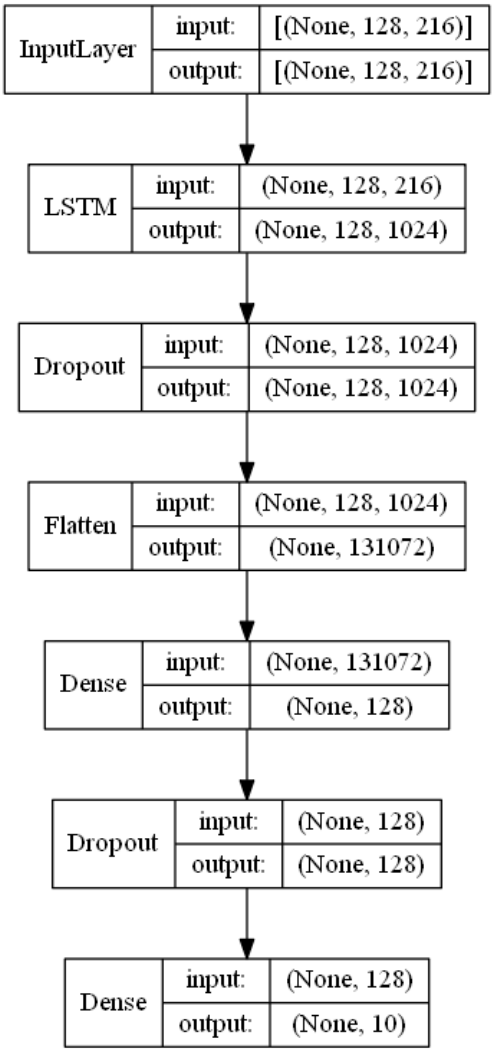


Figure 3.9: Proposed architecture of the LSTM network used for the monophonic classification task.

Neural Network	Feature	Data Augmentation
CNN	Log-Mel Spectrogram	N\A
		Pitch Shifting
		Time Stretching
		WGAN-GP
	MFCCs	N\A
		Pitch Shifting
		Time Stretching
		WGAN-GP
RNN	Log-Mel Spectrogram	N\A
		Pitch Shifting
		Time Stretching
		WGAN-GP
	MFCCs	N\A
		Pitch Shifting
		Time Stretching
		WGAN-GP

Table 3.2: Configurations that were tested for the classification stage of monophonic sounds.

To validate the performance of the classifiers, the technique known as "K-fold Cross-validation" was implemented. Using this technique, the dataset and its corresponding set of target labels are split in k groups or folds. The ESC-10 was already arranged in five uniformly sized folds, but the synthesized data was not, so it was important to ensure that during the distribution of the augmented samples, which depending on the experiment could have been one of the three groups of synthesized data available, the existing classes were evenly distributed among the folds in order to avoid class imbalance. Scikit-Learn library posses a method called "StratifiedKFold()" that provides the indices to split the data in a stratified manner.

The dynamic of K-fold Cross-validation consists in training the model with all the sets except one. This left over set is used as test set. Then, the model is trained again from scratch using other set as test set and the remaining as training set, and so on. With each of the k iterations some metric scores are obtained and averaged to obtain the final scores;

for this task, accuracy, recall, precision and F1-score were the metrics chosen. To avoid overestimated results, it is important to take into account that when working with data augmentation, the synthesized data should only be used for training and not for testing. During the testing only the original data was processed by the model. Additionally, after training and validating a model, confusion matrices of the validation results were displayed to have a better insight of the performance of the tested model.

3.4.5 Polyphonic audio classification

Polyphonic audio classification of environmental sounds was the next stage in this work. In a certain sense it is a similar task to working with monophonic sounds, however, the setup require certain modifications, specially in the architecture of the classifiers. The second part of the methodology is divided in the following sections. First, the construction of a polyphonic audio dataset from a monophonic one is described, then how the data was prepared for the classifiers, the implementation of the classifiers and finally how the training routine, the validation techniques and the evaluation metrics were implemented.

3.4.5.1 Polyphonic audio dataset generation

Each file of the ESC-10 dataset is a recording of an individual class of sound. To generate a polyphonic audio dataset for the task of Audio Tagging (multi-label classification) from this dataset, a proposed procedure from a different investigation was implemented [21]. The steps are mentioned below.

First, as it was mentioned previously, the ESC-10 dataset already comes divided in five different folds, however, for the task of Audio Tagging the number of folds was incremented from five to ten in order to decrease a possible statistical bias during the training of the different models. To create the extra folds, half of the files of each class from each fold were chosen randomly and assigned to a new fold. This ensure that the distribution of classes remained balanced. This results in each fold having four different files per class, giving a total of 40 files per fold.

Once the folds were prepared, each file of each fold was overlapped with one file of each of the other nine classes pertaining to the same fold at a sampling rate of 44100 Hz, giving a total of 45 files (Figure 3.10). Since there are four files per class in each fold repeating this process creates 180 files for one fold. Finally, the process is repeated for the remaining folds,

bringing a total of 1800 files. This process can be represented by the following equation (eq 3.1):

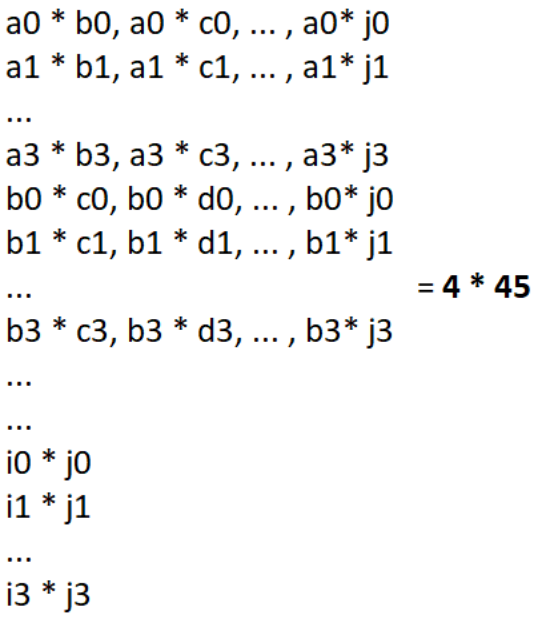


Figure 3.10: Algorithm used to generate the database of polyphonic sounds from the ESC-10 dataset.

$$kn \sum_{i=1}^{m-1} m = 1800 \tag{3.1}$$

being n the number of samples per class per fold, m the number of classes and k the number of folds. Also, since the duration of the original files is five seconds, the resultant combined files will have the same length.

During the process of generating the polyphonic dataset, a Pandas dataframe is created with the purpose of preserving relevant information of each combined file (Figure 3.2). To assign the name of the combined file four digits are taken into account, for example, one possible name for a file could be "2_5_4_1.wav". The first digit represents the class id of the first audio, the second digit the class id of the second audio, the third digit the fold to which this new audio belong and the last digit indicates the number of instance of that combination of

classes. The dataframe is also exported as a .csv file.

Additionally, three more datasets of polyphonic sounds were generated using the same algorithm with the same files that were created by implementing the three methods of data augmentation that were used during the monophonic classification task (WGAN-GP, pitch shifting and time stretching). This additional polyphonic files (1600 files per augmentation method) were created separately from the original ESC-10 files, which means there are no mixes between the original and the synthesized files. Merging each of this datasets with the one made from original files gives three datasets consisting of 3600 files.

Unnamed: 0	name	audio_1	audio_2	clase_1	clase_2	fold
0	4_0_1_0	1-100032-A-0.wav	1-116765-A-41.wav	dog	chainsaw	1
1	4_2_1_0	1-100032-A-0.wav	1-17150-A-12.wav	dog	crackling_fire	1
2	4_5_1_0	1-100032-A-0.wav	1-172649-A-40.wav	dog	helicopter	1
3	4_6_1_0	1-100032-A-0.wav	1-17367-A-10.wav	dog	rain	1
4	4_3_1_0	1-100032-A-0.wav	1-187207-A-20.wav	dog	crying_baby	1
...	...	...	...	...	...	...
1795	7_8_5_3	5-200334-B-1.wav	5-208810-B-11.wav	rooster	sea_waves	5
1796	7_8_5_4	5-200339-A-1.wav	5-213077-A-11.wav	rooster	sea_waves	5
1797	7_8_5_5	5-233160-A-1.wav	5-219379-A-11.wav	rooster	sea_waves	5
1798	7_8_5_6	5-234879-A-1.wav	5-219379-B-11.wav	rooster	sea_waves	5
1799	7_8_5_7	5-234879-B-1.wav	5-219379-C-11.wav	rooster	sea_waves	5

1800 rows × 7 columns

Figure 3.11: Pandas dataframe generated during the process of creating the polyphonic audio dataset. The columns represent the id ("Unnamed: 0"), the name of the generated audio ("name"), the name of the original files ("audio_1" and "audio_2"), the id of the classes of the original files ("clase_1" and "clase_2") and the fold ("fold").

3.4.5.2 Preprocessing and feature extraction

Once the polyphonic audio datasets were created, the same preprocessing algorithms and feature extraction techniques explained for the mobophonic approach were applied. The preprocessing steps include the removal of "silences", the downsampling from 44100 Hz to

22050 Hz and the homogenization of the duration of each audio signal.

About the extraction of features, both log-mel spectrograms and MFCCs were obtained with the same functions and the same parameter values used in the monophonic classification task.

3.4.5.3 Classification

The classification of the features extracted from the different custom datasets was done by a covolutional neural network and a LSTM network similar in their architecture to the ones that were used in the monophonic classification. The main modification to the architecture of each network was the use of a sigmoid activation function in the last layer instead of a softmax function and the use of binary cross-entropy as loss function. This is a requisite for multi-label classification in order to have independent probabilities for each of the classes, since in this case the classes aren't mutually exclusive. This could be seen as if the multi-label classification problem turned into 10 separate binary classification problems.

Regularization layers were also applied for each of the networks in order to decrease a probable overfitting and overconfidence of the models. For the CNN, a dropout layer was placed after the last max pooling layer with a value of 0.5 for the fraction of input units to drop. Also, batch normalization layers were placed after each convolutional layer. On the other hand, for the LSTM, in addition of using dropout layers, Label Smoothing method with a value of 0.2 was applied each time the binary cross-entropy loss was computed. Label Smoothing reduces the gap between the predicted values so that the largest values do not become much bigger than the rest, i.e., reduces the overconfidence of the model. The architecture of the proposed neural networks are shown in Figures and Figures.

Finally, each possible model (combination of neural network, feature and dataset) was trained for 50 epochs with a batch size of 32. On the other hand, in order to maximize the performance of each model, the value of the learning rate was carefully tuned for each possible combination of neural network and feature. To validate their performance again K-fold Cross-Validation technique was implemented as in the monophonic classification task but this time with a value of 10 for K. Also, as it was previously stated, for each iteration only features that come from the original dataset were used during the testing to avoid any overestimation of the classifier's performance.

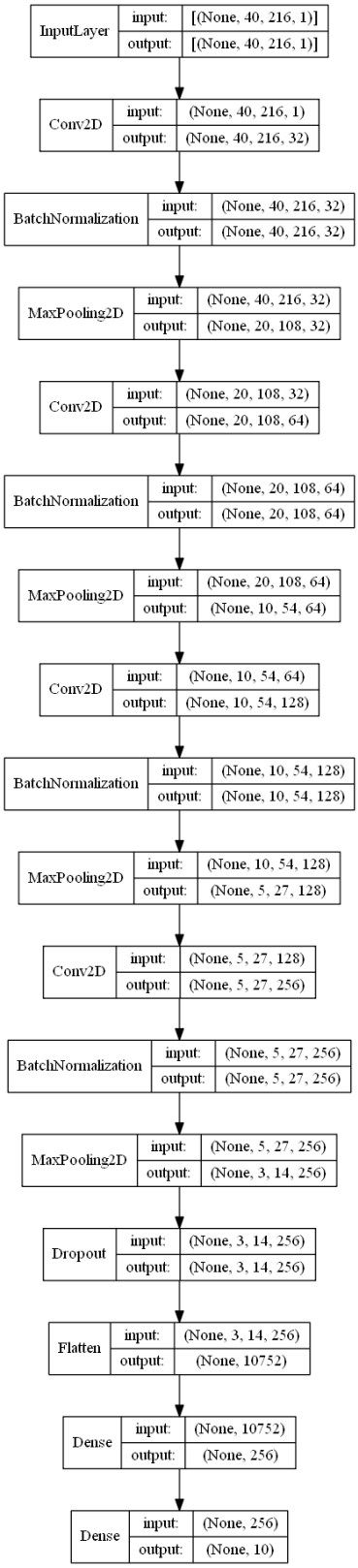


Figure 3.12: Proposed architecture of the CNN used for the polyphonic classification task.

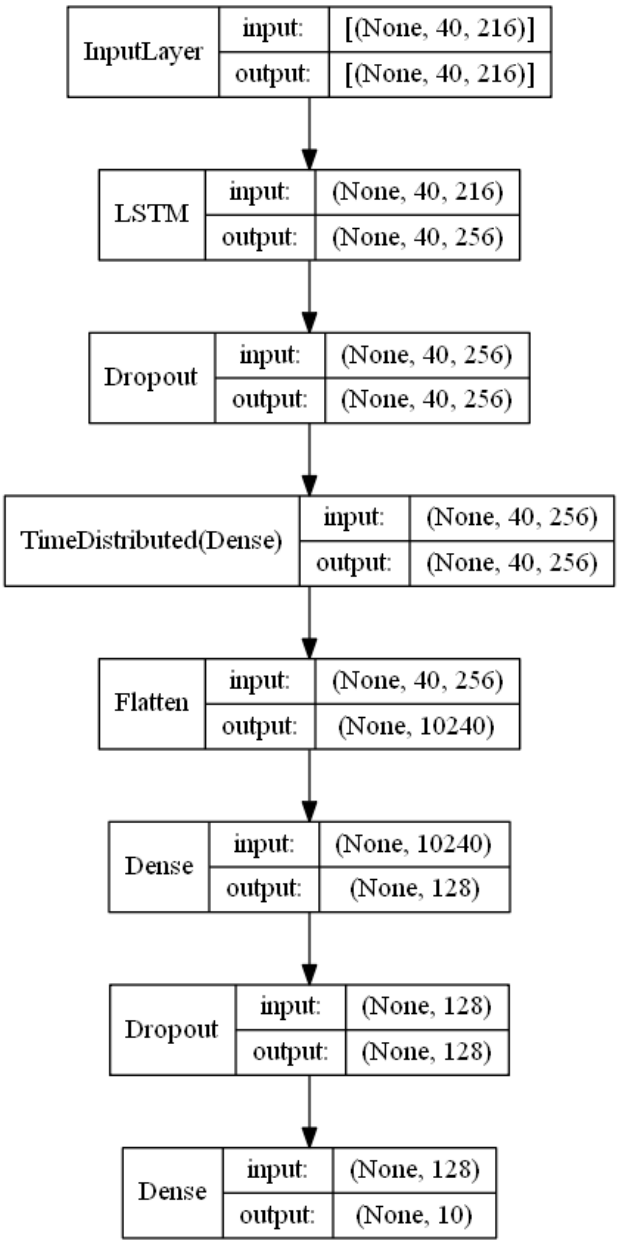


Figure 3.13: Proposed architecture of the LSTM network used for the polyphonic classification task.

3.4.5.4 Evaluation

Regarding the evaluation metrics for this task, Hamming Loss, global accuracy, precision, recall and F1-score were obtained for each iteration of the K-fold Cross-Validation using a threshold value of 0.55 which favours precision over recall. Once the training finished, the micro-average of all the evaluation metrics was computed. Micro-average was chosen as aggregation method to take into account the contribution of each label to the final value.

Precision-recall curves were also generated for each configuration of neural network, feature and dataset to have a better insight of the behaviour of each model. For each curve, average precision (the area under the curve) was obtained. Finally, when all the different configurations were tested, precision-recall curves for each class were generated, including one for the micro-average over all classes (Figure 4.73).

CHAPTER 4

Results and Discussion

In this chapter the results obtained from testing different combinations of neural network, features and dataset for both the monophonic and polyphonic audio classification tasks are presented. Finally, the interpretation of the results and some comparatives with other researches are presented.

4.1 Results

The information is presented so that first the monophonic classification task results are shown, starting with the results obtained by combining the convolutional neural network with the different available datasets and features. Then, the results obtained by using the LSTM network with the same combinations of network and datasets are presented. The visual material that is presented correspond to graphs of the accuracy and loss obtained during the training using the method of K-Fold Cross-Validation with a k=5. Normalized and unnormalized confusion matrices of each experiment are shown to give a visual insight of the performance of each model. For each experiment, evaluation metrics such as accuracy, precision, recall and f1-score are mentioned in the caption located below the confusion matrices, however, all the values of all experiments are condensed in table 4.1.

For the polyphonic classification task, the order of how the experiments are presented is the same as in the previous task, however, the visual material presented is different. For each experiment, precision and loss graphs of the training using a k=10 for the K-Fold Cross-Validation method are shown, followed by precision-recall curves and confusion matrices per class. Finally, evaluation metrics per class and its micro average are presented in a table.

All the results of this task are summarized in table 4.2.

For the sake of not confusing the reader, the graphs and confusion matrices that are shown are the ones obtained from running the original dataset, the augmented dataset with the samples generated by the generative adversarial network and the pitch shifting augmentation method. The reason the latter method's results are included is because it is considered to be the one that obtained the best results.

4.1.1 Monophonic dataset classification

Data Augmentation: N/A,Model: CNN, Feature: Log-Mel Spectrograms

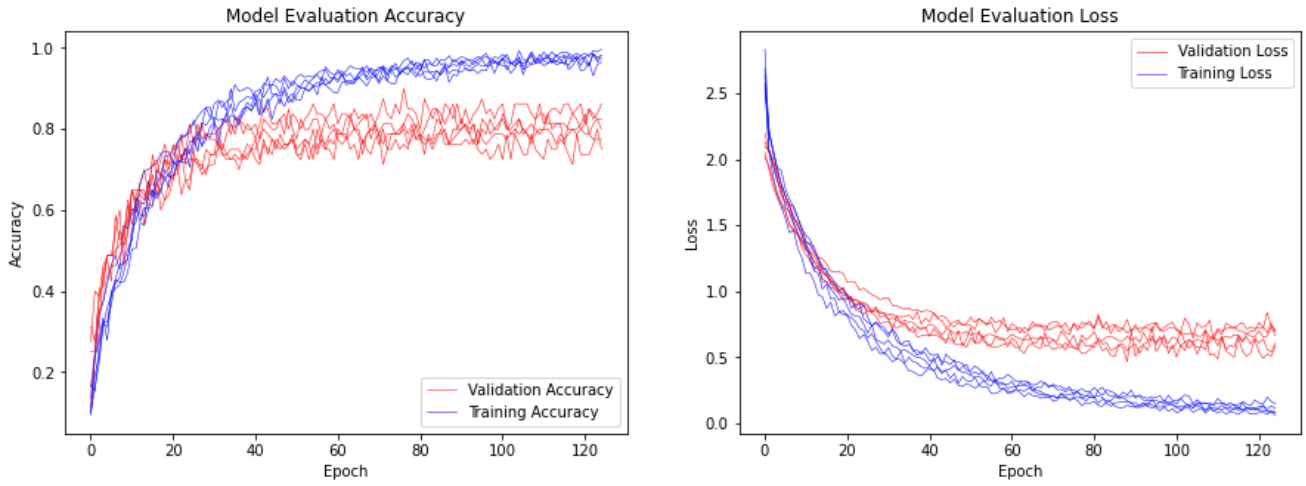
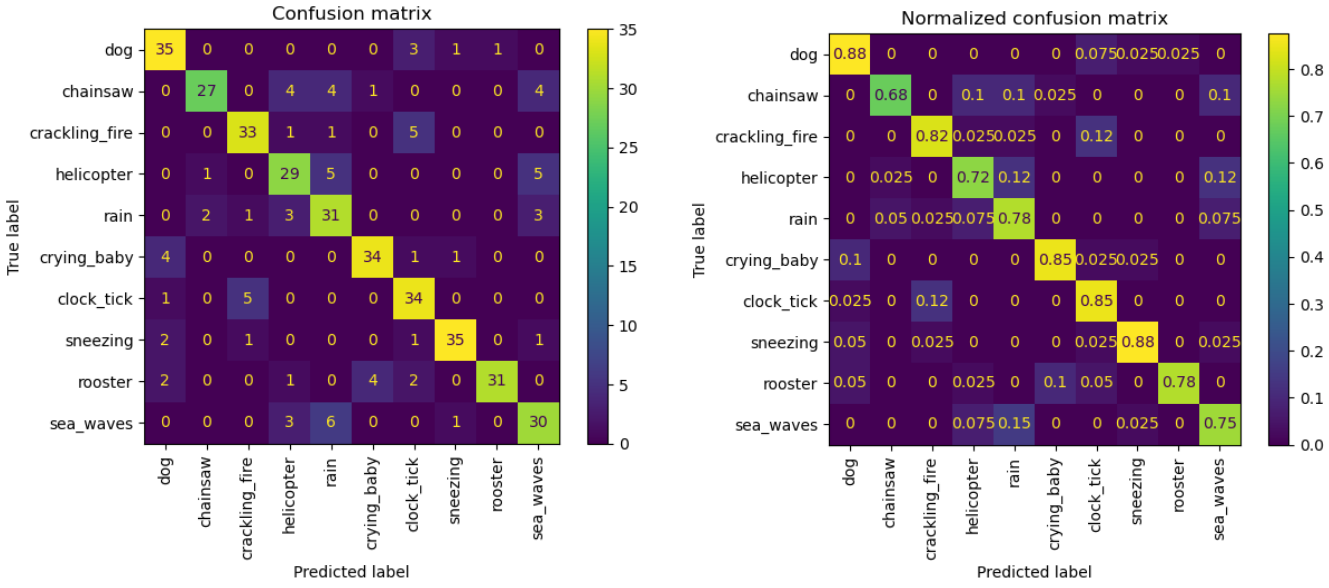


Figure 4.1: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 80.24% accuracy, 83.02% precision, 80.25% recall and 79.62% F1-Score.



(a) (b)

Figure 4.2: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: WGAN-GP, Model: CNN, Feature: Log-Mel Spectrograms

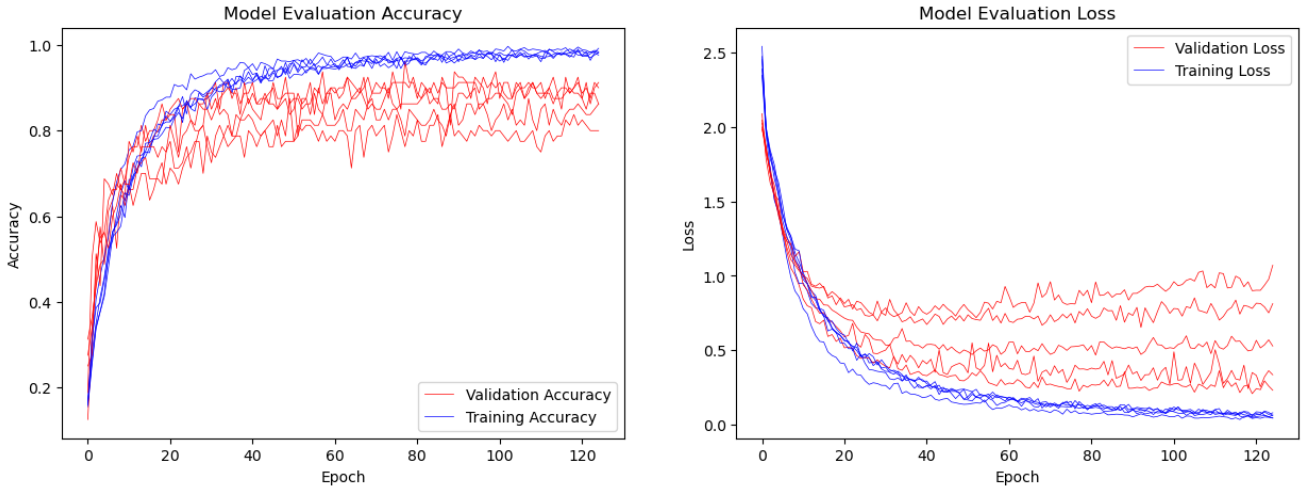
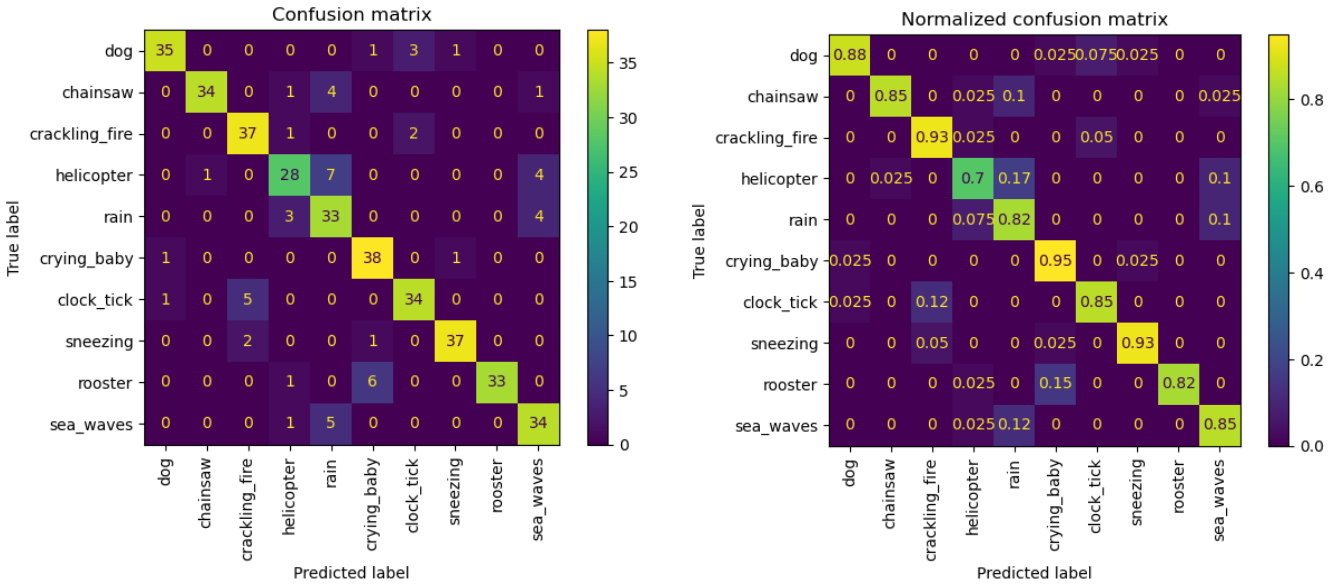


Figure 4.3: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 86.75% accuracy, 88.18% precision, 86.75% recall and 86.49% F1-Score.



(a) (b)

Figure 4.4: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: Pitch Shifting, Model: CNN, Feature: Log-Mel Spectrograms

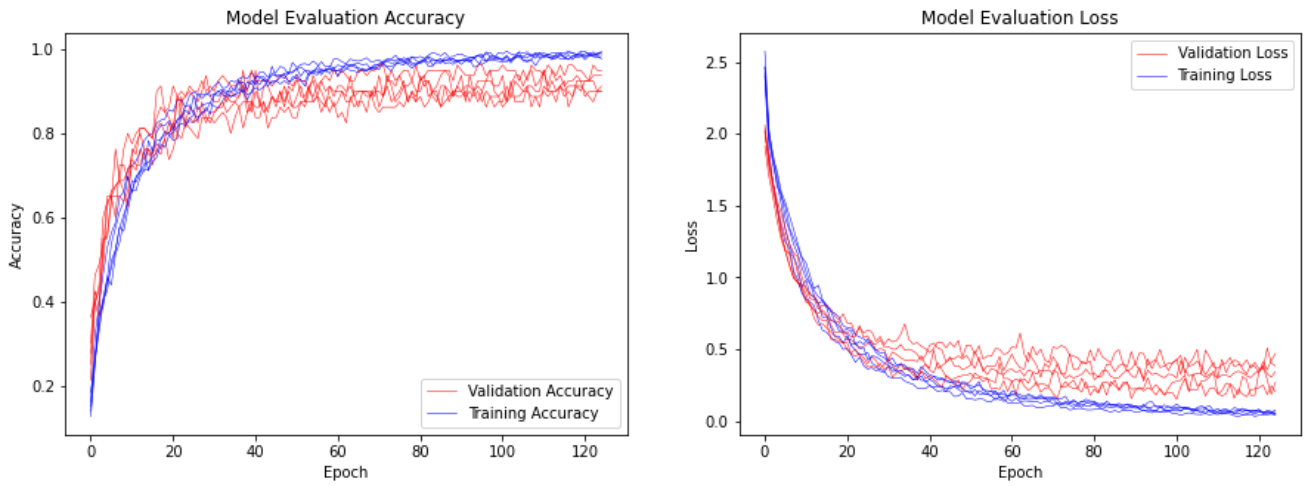
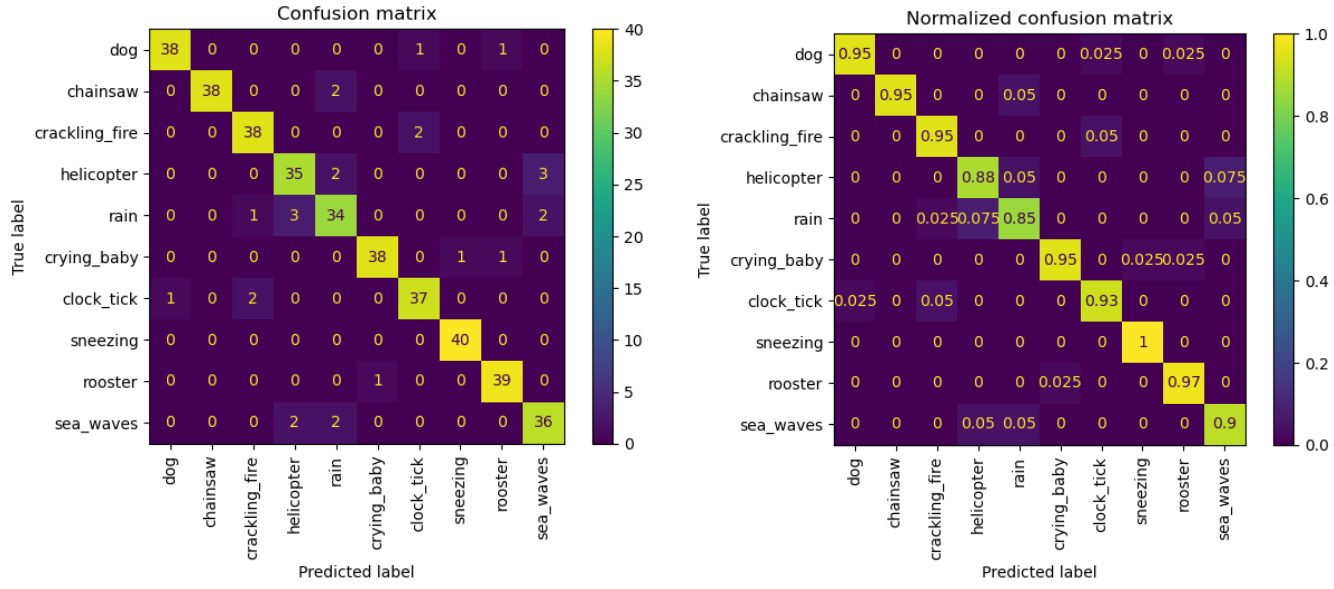


Figure 4.5: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 92% accuracy, 93.48% precision, 92% recall and 91.62% F1-Score.



(a) (b)

Figure 4.6: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: N/A, Model: CNN, Feature: MFCCs

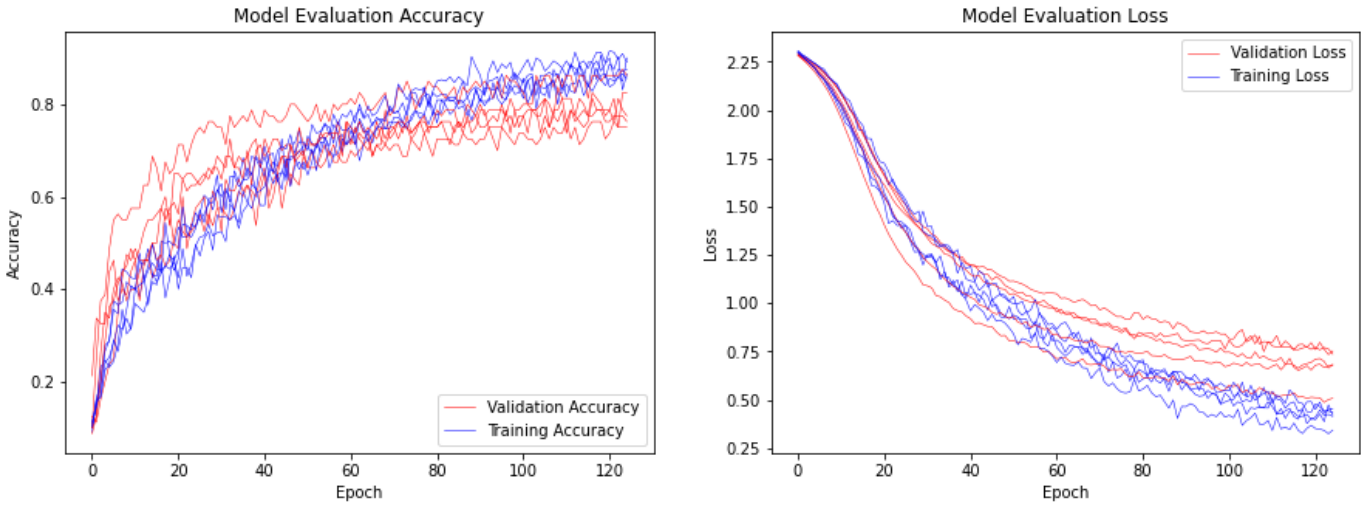
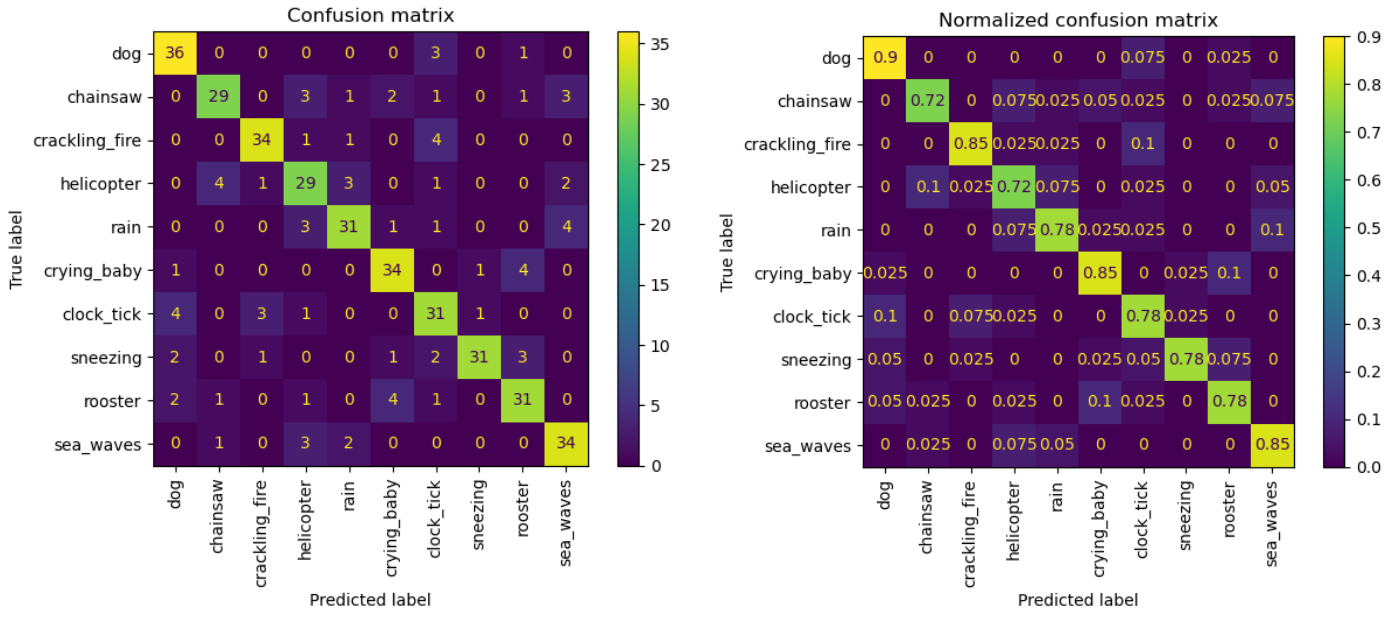


Figure 4.7: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 79.5% accuracy, 81.67% precision, 79.5% recall and 79.18% F1-Score.



(a) (b)

Figure 4.8: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: WGAN-GP, Model: CNN, Feature: MFCCs

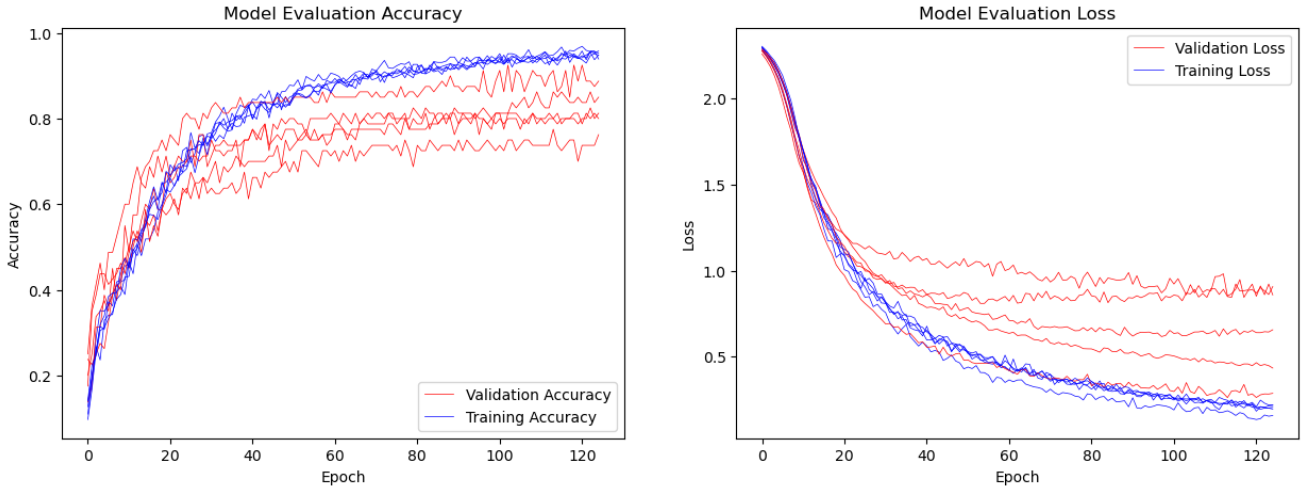
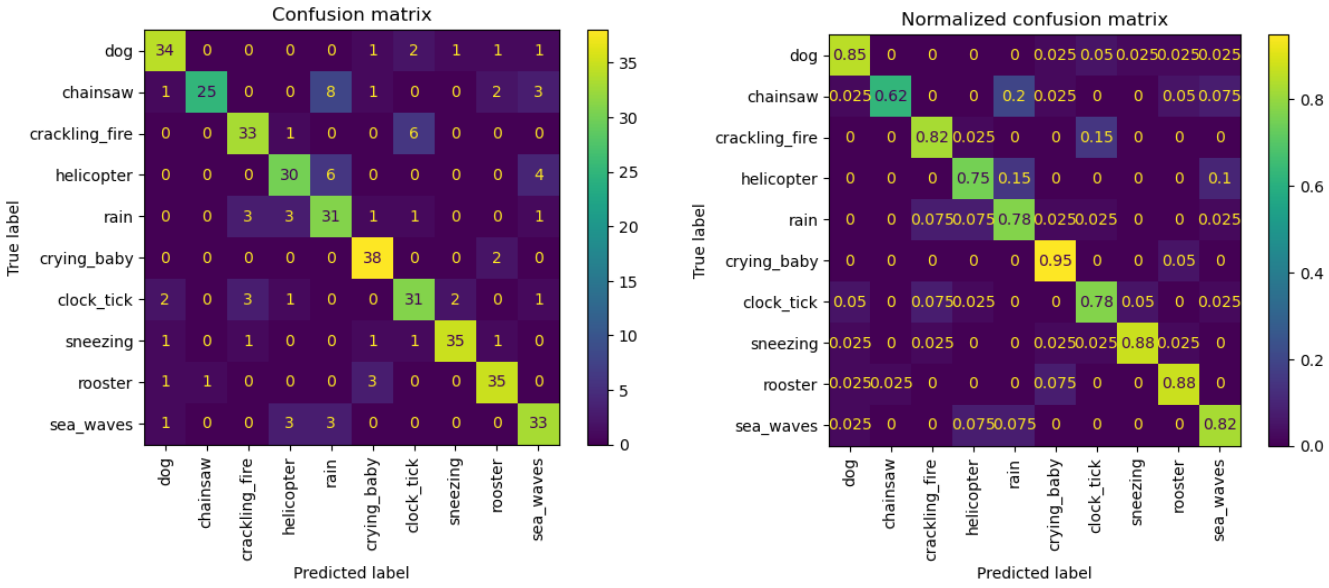


Figure 4.9: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 82.25% accuracy, 83.09% precision, 82.25% recall and 81.68% F1-Score.



(a) (b)

Figure 4.10: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: Pitch Shifting, Model: CNN, Feature: MFCCs

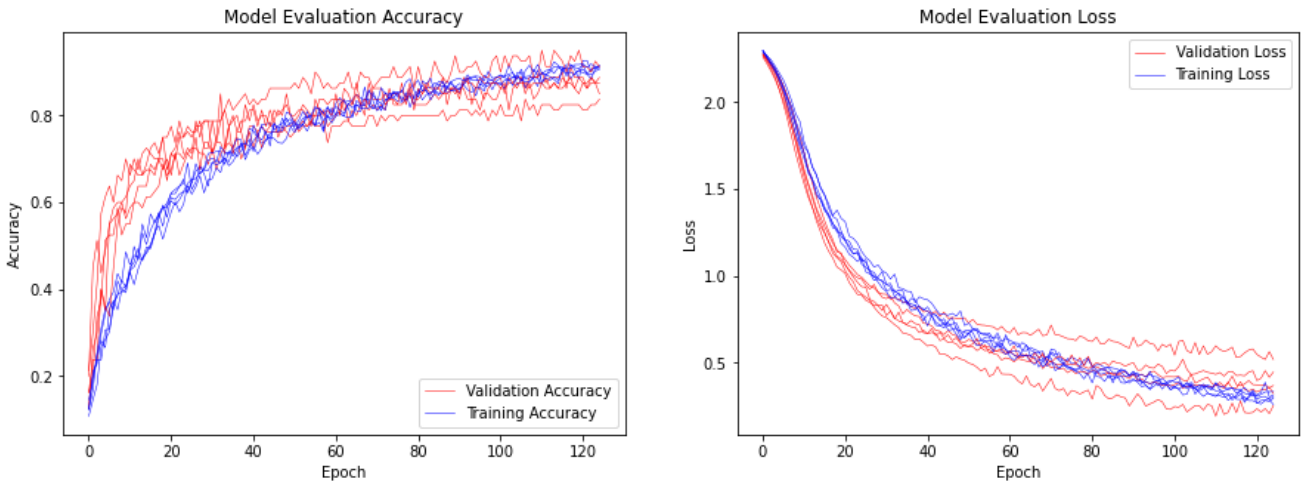
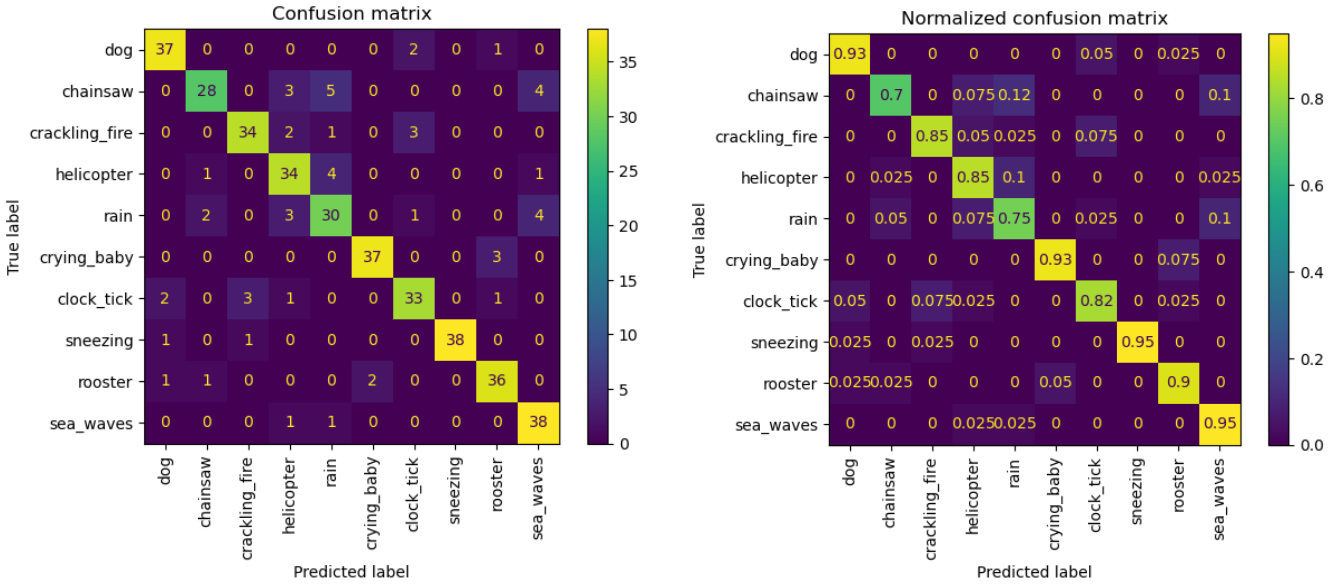


Figure 4.11: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 87.25% accuracy, 88.68% precision, 87.25% recall and 87.01% F1-Score.



(a) (b)

Figure 4.12: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: N/A, Model: LSTM, Feature: Log-Mel Spectrograms

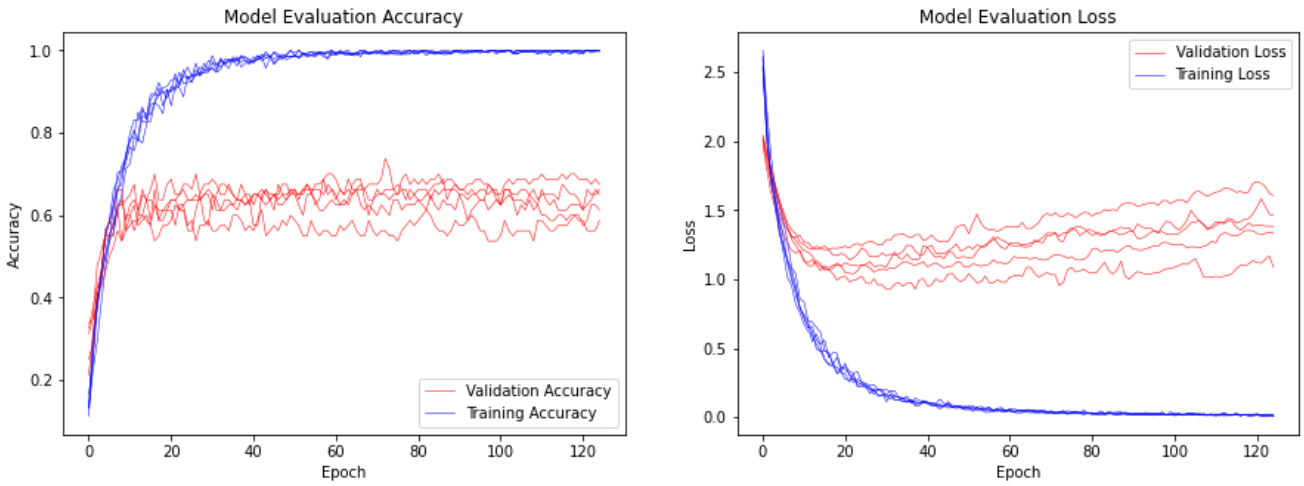
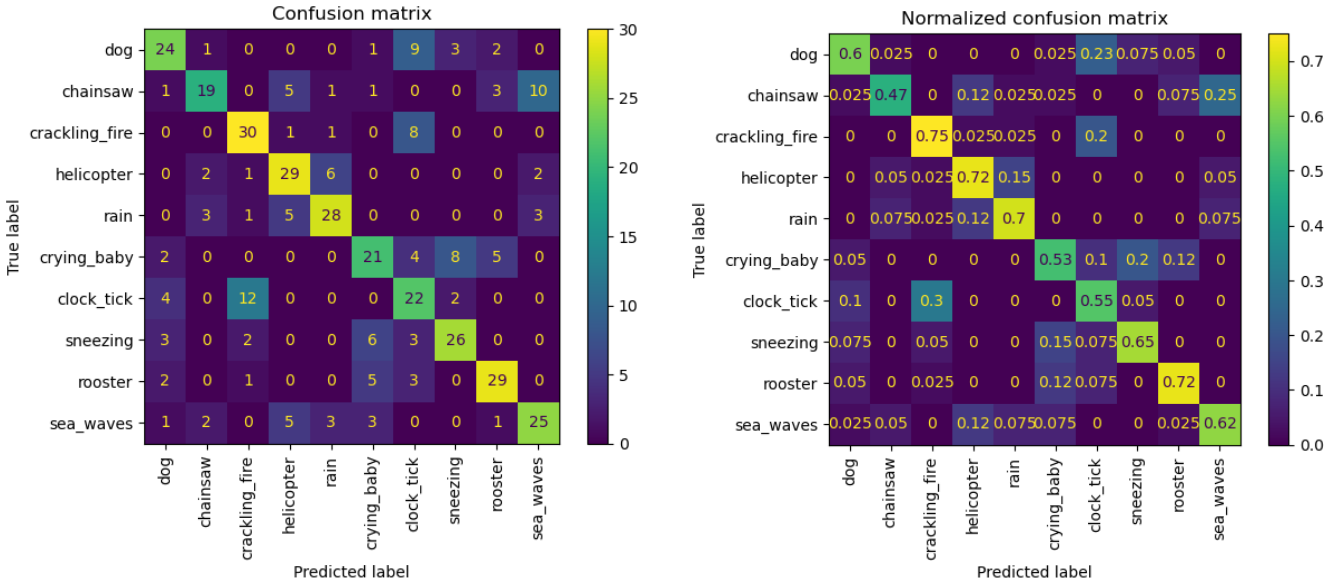


Figure 4.13: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 63.75% accuracy, 66.03% precision, 63.75% recall and 63.06% F1-Score.



(a) (b)

Figure 4.14: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: WGAN-GP, Model: LSTM, Feature: Log-Mel Spectrograms

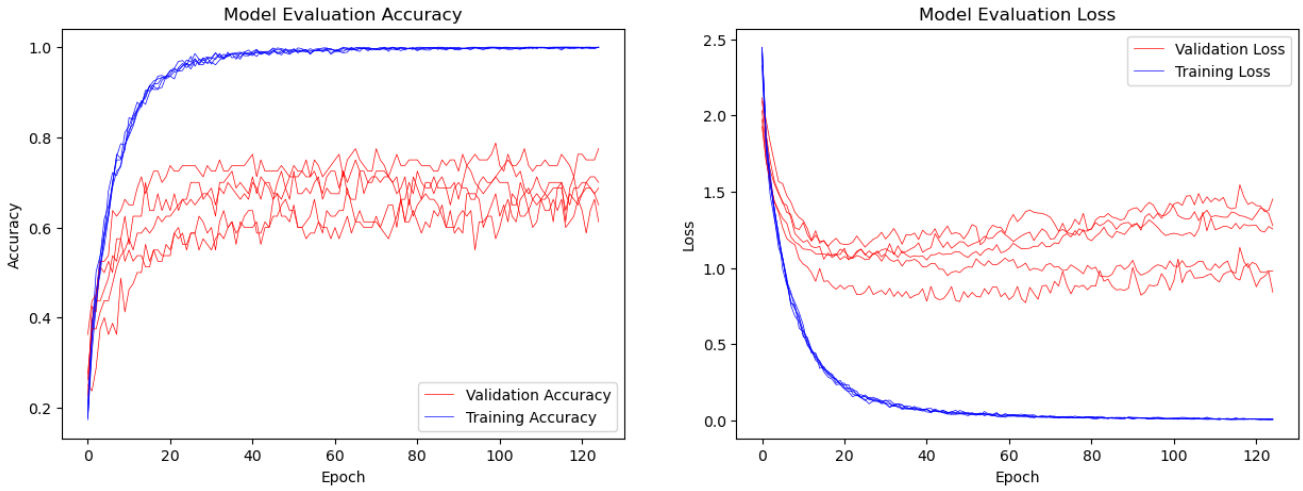
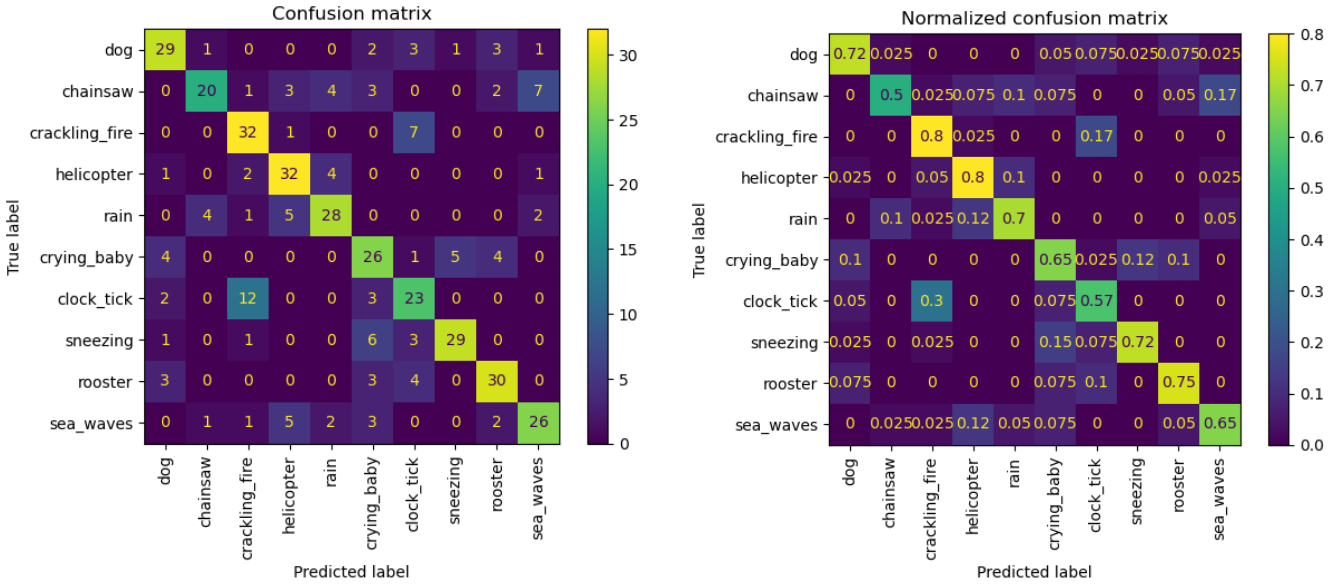


Figure 4.15: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 68.5% accuracy, 70.76% precision, 68.5% recall and 68.16% F1-Score.



(a) (b)

Figure 4.16: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: Pitch Shifting, Model: LSTM, Feature: Log-Mel Spectrograms

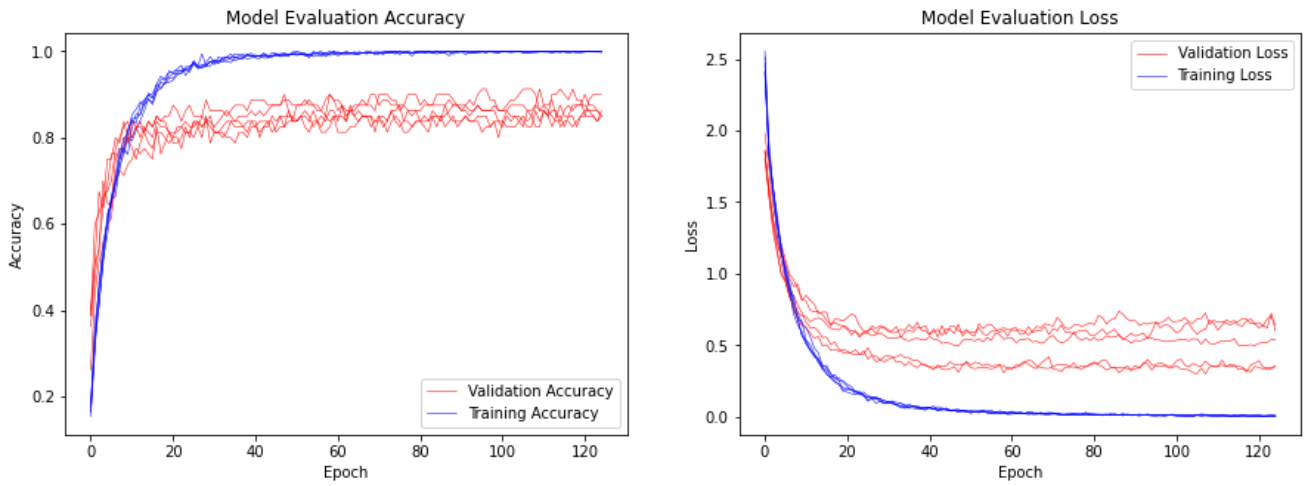
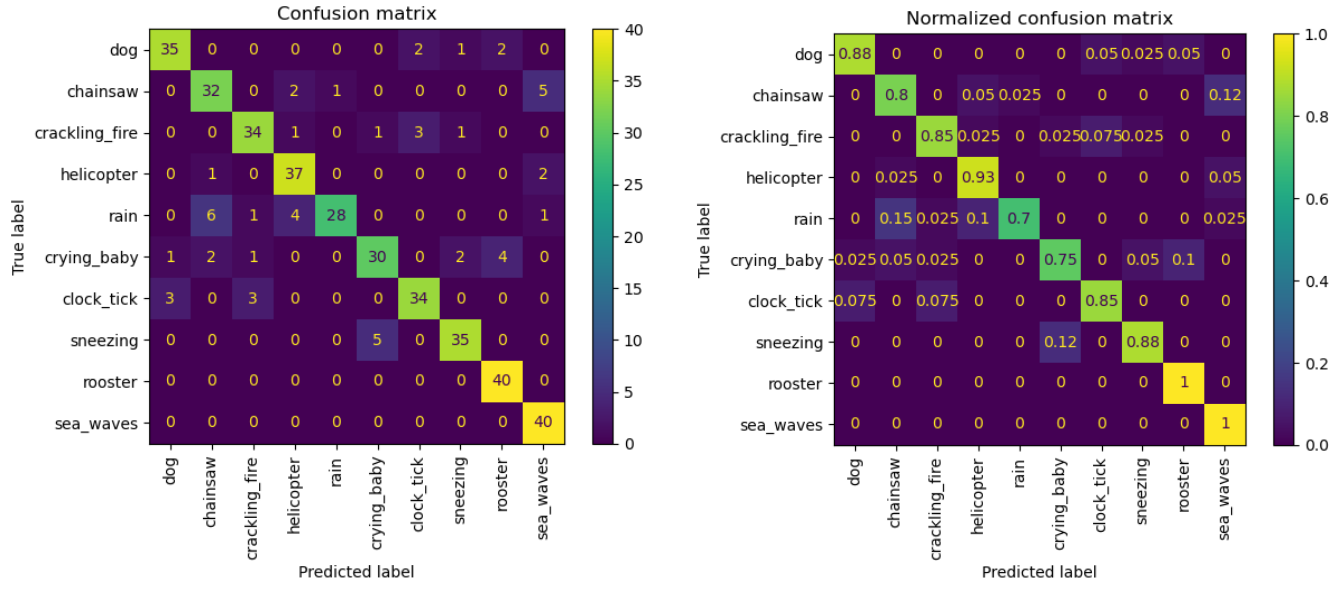


Figure 4.17: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 86.25% accuracy, 87.48% precision, 86.25% recall and 86.03% F1-Score.



(a) (b)

Figure 4.18: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: N/A, Model: LSTM, Feature: MFCCs

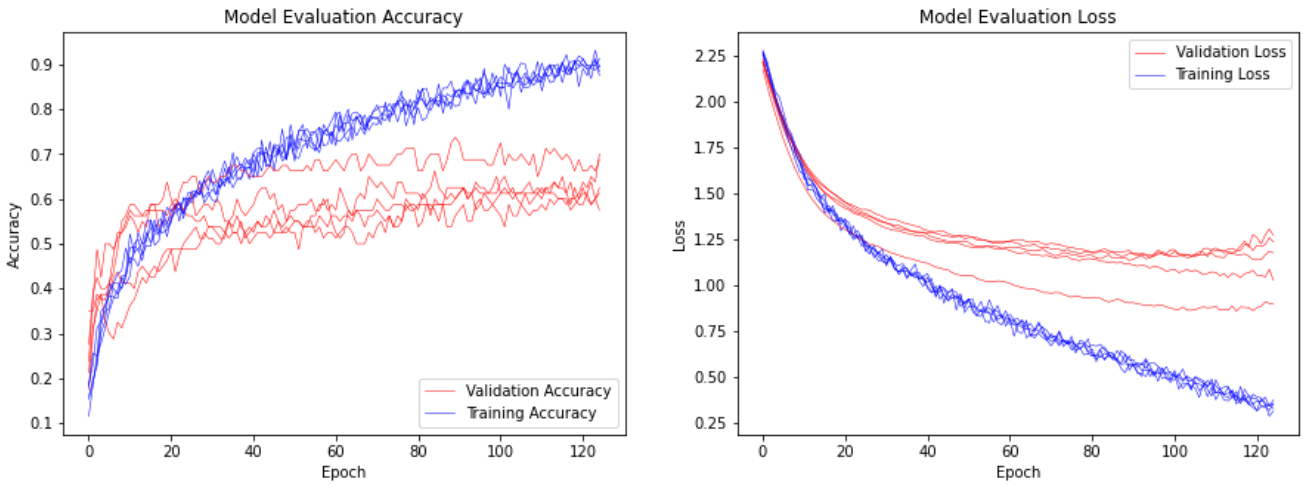
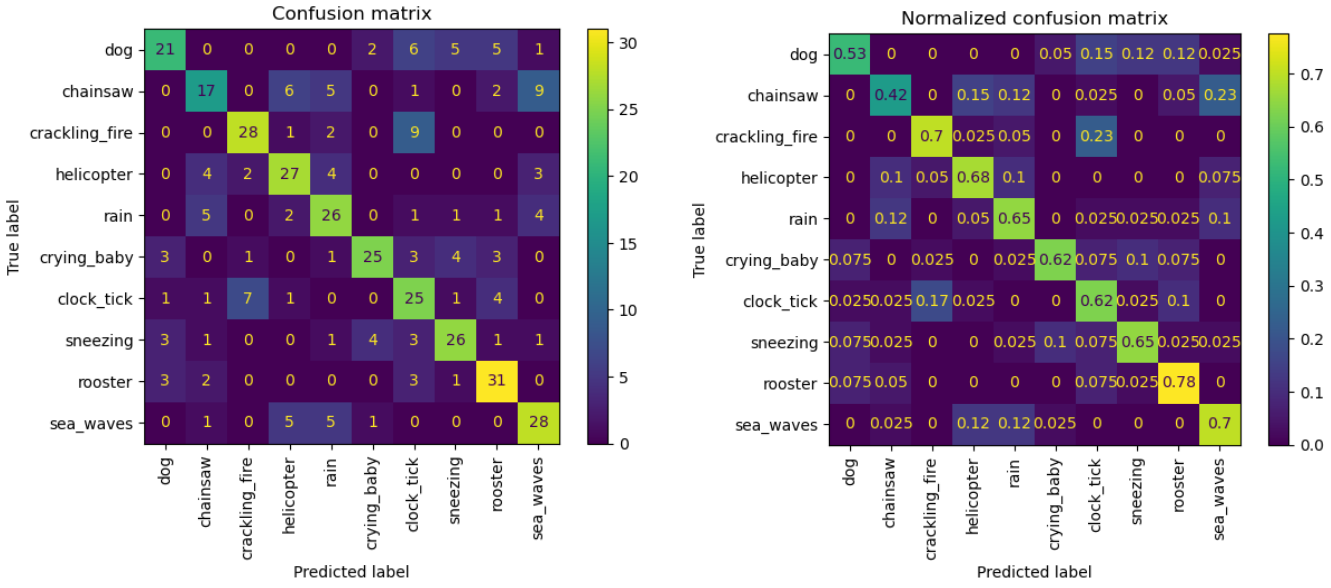


Figure 4.19: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 64% accuracy, 65.87% precision, 64% recall and 63.09% F1-Score.



(a) (b)

Figure 4.20: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: WGAN-GP, Model: LSTM, Feature: MFCCs

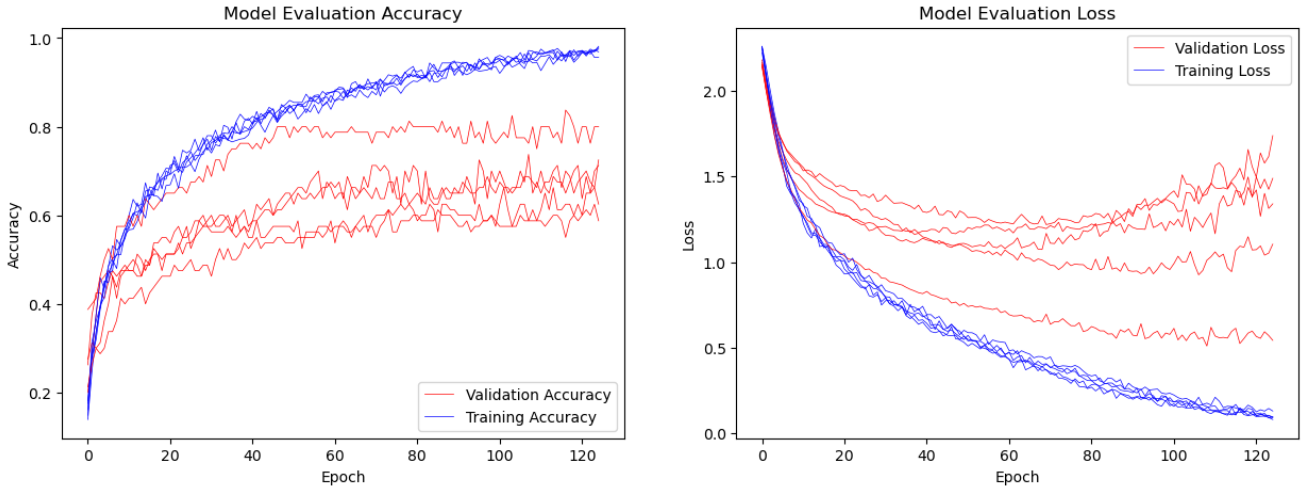
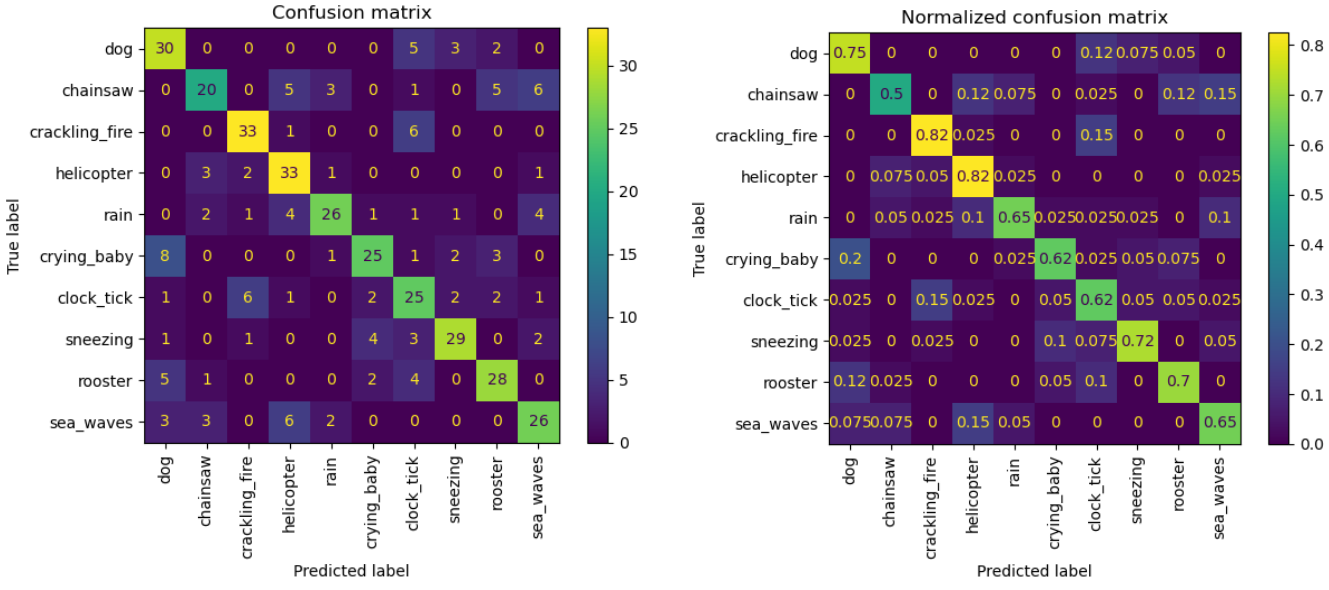


Figure 4.21: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 69% accuracy, 71.19% precision, 69% recall and 68.32% F1-Score.



(a) (b)

Figure 4.22: a) Unnormalized and b) normalized confusion matrices.

Data Augmentation: Pitch Shifting, Model: LSTM, Feature: MFCCs

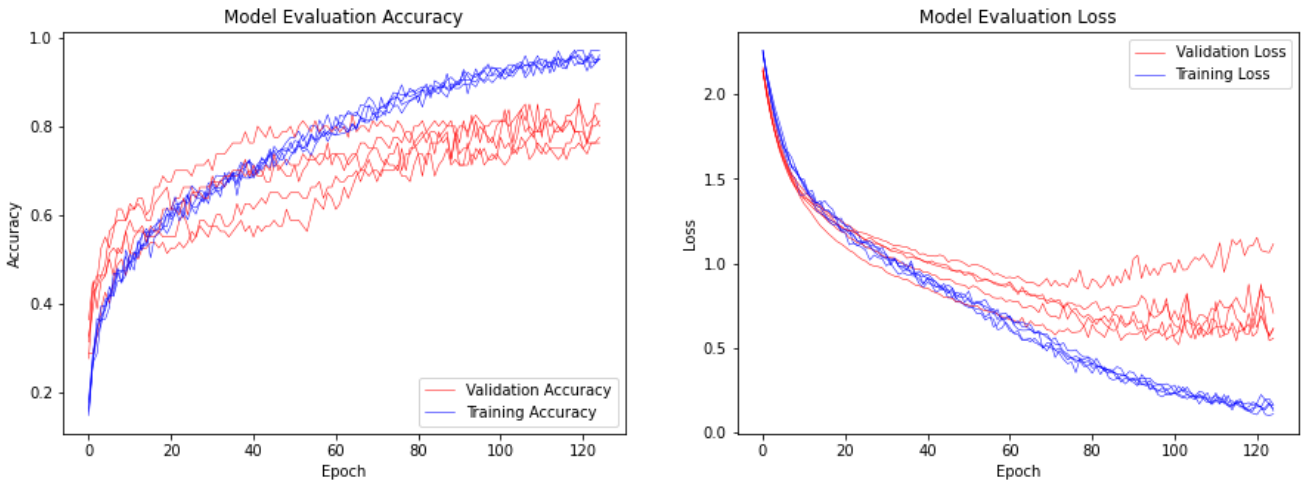
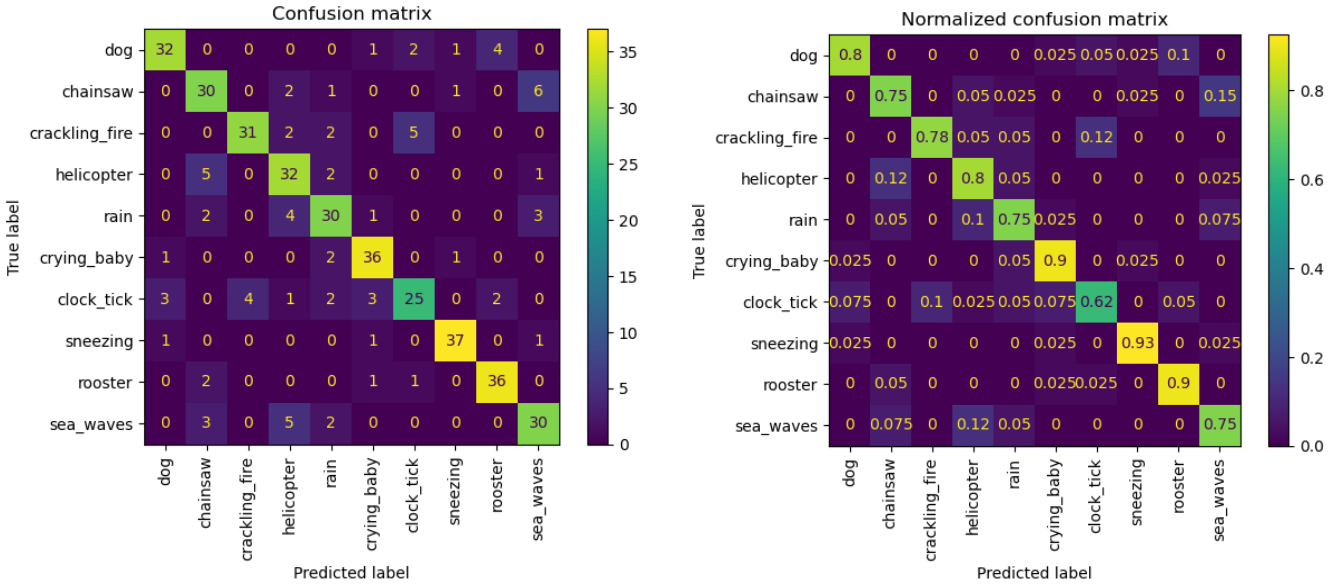


Figure 4.23: Accuracy and loss graph of the proposed model. After 125 epochs the model scored 80% accuracy, 81.42% precision, 80% recall and 79.02% F1-Score.



(a) (b)

Figure 4.24: a) Unnormalized and b) normalized confusion matrices.

Neural Network + Feature	**Augmentation**	**No. Files**	**Acc.**	**Prec.**	**Recall**	**F1-Score**	**Tr. Loss**
CNN + Log-Mel Spectrogram	N\A	400	80.24%	83.02%	80.25%	79.62%	0.65
	WGAN-GP	800	86.75%	88.18%	86.75%	86.49%	0.6
	Time Stretching	800	88.25%	89.84%	88.25%	88.18%	0.36
	Pitch Shifting	800	**92%**	**93.48%**	**92%**	**91.62%**	0.33
CNN + MFCCs	N\A	400	79.5%	81.67%	79.5%	79.18%	0.67
	WGAN-GP	800	82.25%	83.09%	82.25%	81.68%	0.63
	Time Stretching	800	85.25%	87.05%	85.25%	84.85%	0.45
	Pitch Shifting	800	**87.25%**	**88.68%**	**87.25%**	**87.01%**	0.38
LSTM + Log-Mel Spectrogram	N\A	400	63.75%	66.03%	63.75%	63.06%	1.38
	WGAN-GP	800	68.5%	70.76%	68.5%	68.16%	1.16
	Time Stretching	800	71.25%	74.86%	71.25%	70.7%	1.15
	Pitch Shifting	800	**86.25%**	**87.48%**	**86.25%**	**86.03%**	0.5
LSTM + MFCCs	N\A	400	64%	65.87%	64%	63.09%	1.12
	WGAN-GP	800	69%	71.19%	69%	68.32%	1.24
	Time Stretching	800	67.5%	70.49%	67.5%	67.19%	0.92
	Pitch Shifting	800	**80%**	**81.42%**	**80%**	**79.02%**	0.72

Table 4.1: Results of the monophonic stage. The best results of each configuration (Neural Network + feature) are highlighted in bold.

4.1.2 Polyphonic dataset classification

Data Augmentation: N/A, Model: CNN, Feature: Log-Mel Spectrograms

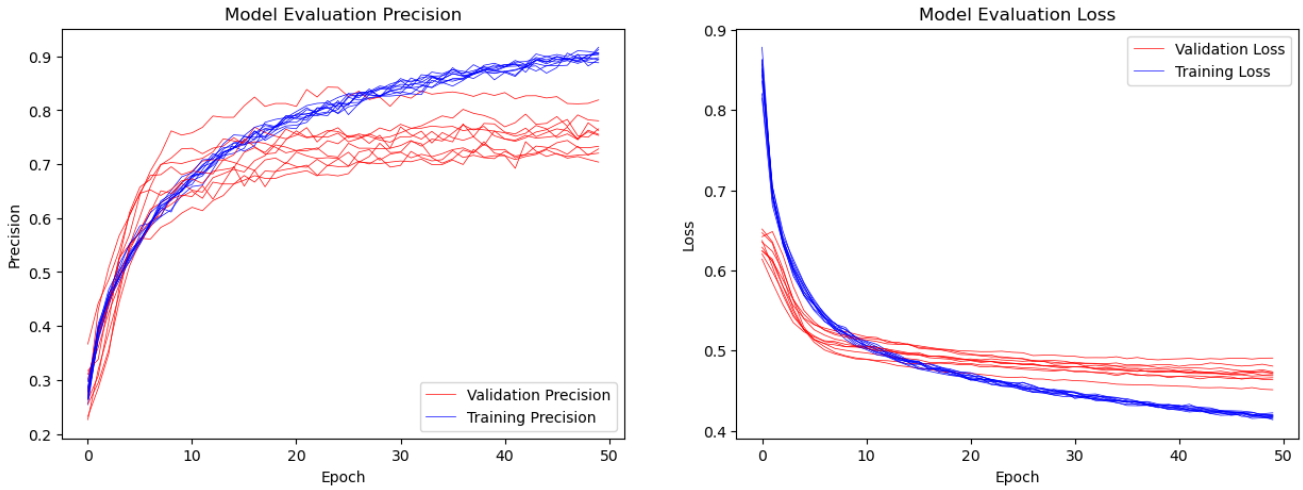


Figure 4.25: Precision and loss graphs obtained from evaluating the proposed convolutional neural network with a 10-fold cross validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset without data augmentation.

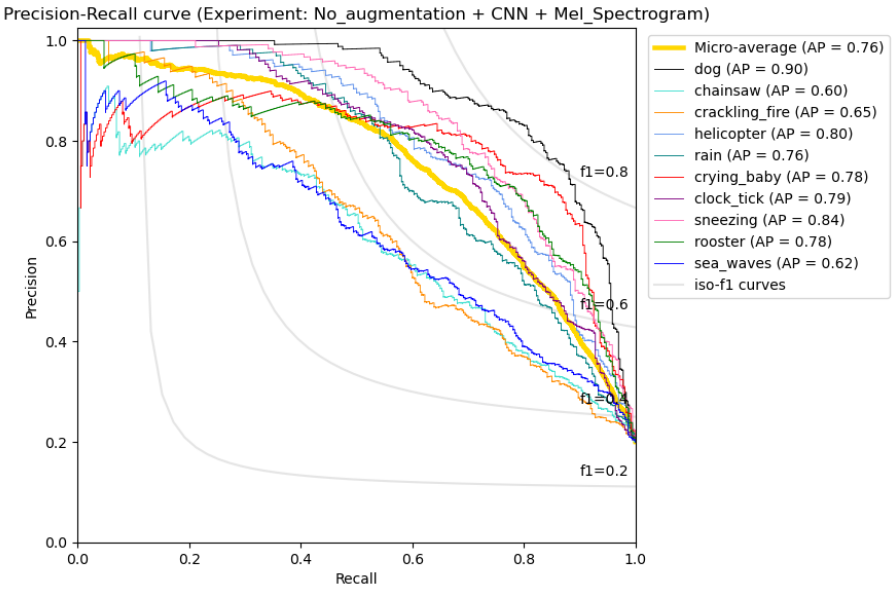


Figure 4.26: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.76 was obtained. Dog class obtained the highest value (AP = 0.9) while the worst was obtained by the chainsaw class (AP = 0.6).

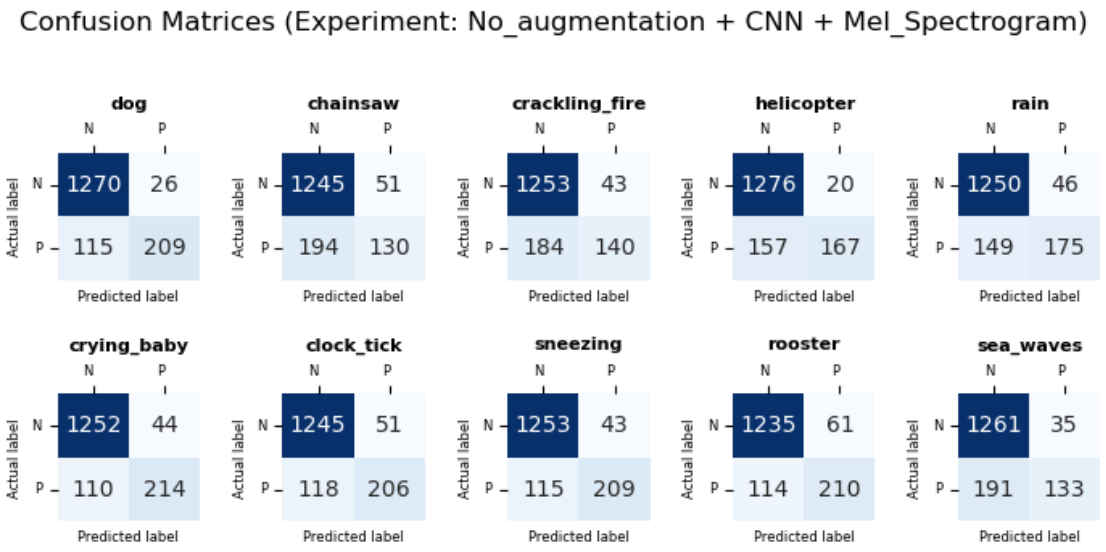


Figure 4.27: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	91.29	63.89	72.18
chainsaw	75.49	40.56	51.42
crackling_fire	77.13	41.39	52.37
helicopter	90.8	50.0	62.24
rain	83.49	54.72	64.92
crying_baby	83.45	69.17	74.11
clock_tick	82.57	62.5	68.52
sneezing	84.95	66.94	73.56
rooster	82.49	64.72	71.07
sea_waves	77.7	38.61	50.4
micro avg	81.14	55.25	65.7

Average Hamming Loss: 0.12
Average global accuracy (exact match): 41.11%
Average loss: 0.47

Figure 4.28: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed convolutional neural network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: WGAN-GP, Model: CNN, Feature: Log-Mel Spectrograms

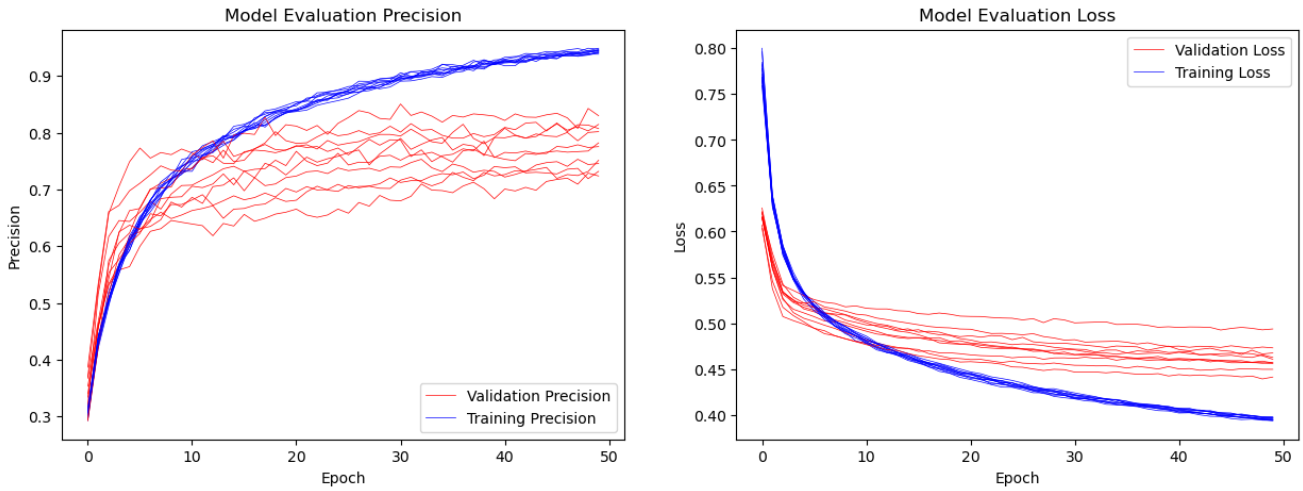


Figure 4.29: Precision and loss graphs obtained from evaluating the proposed convolutional neural network with a 10-fold cross validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the WGAN-GP augmented files.

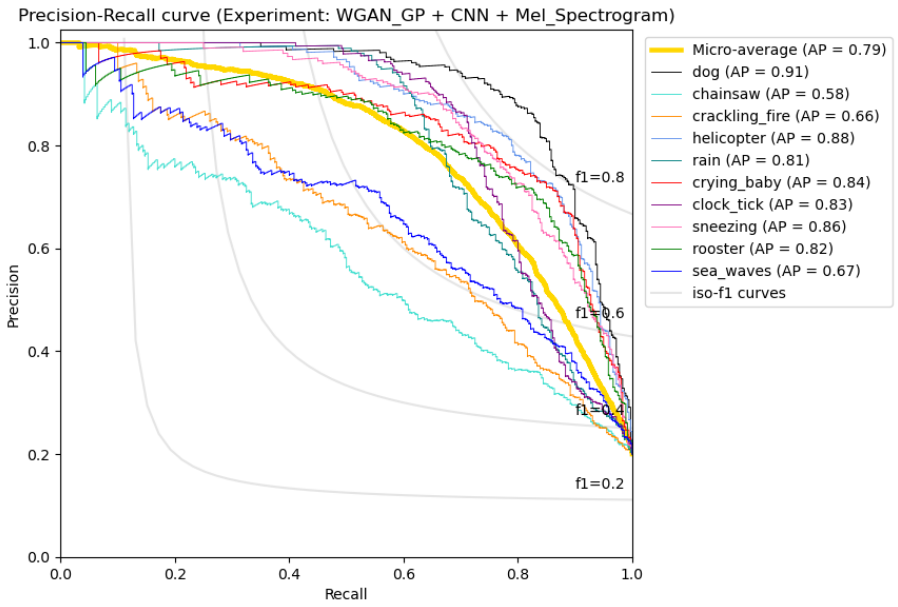


Figure 4.30: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.79 was obtained. Dog class obtained the highest value (AP = 0.91) while the worst was obtained by the chainsaw class (AP = 0.58).

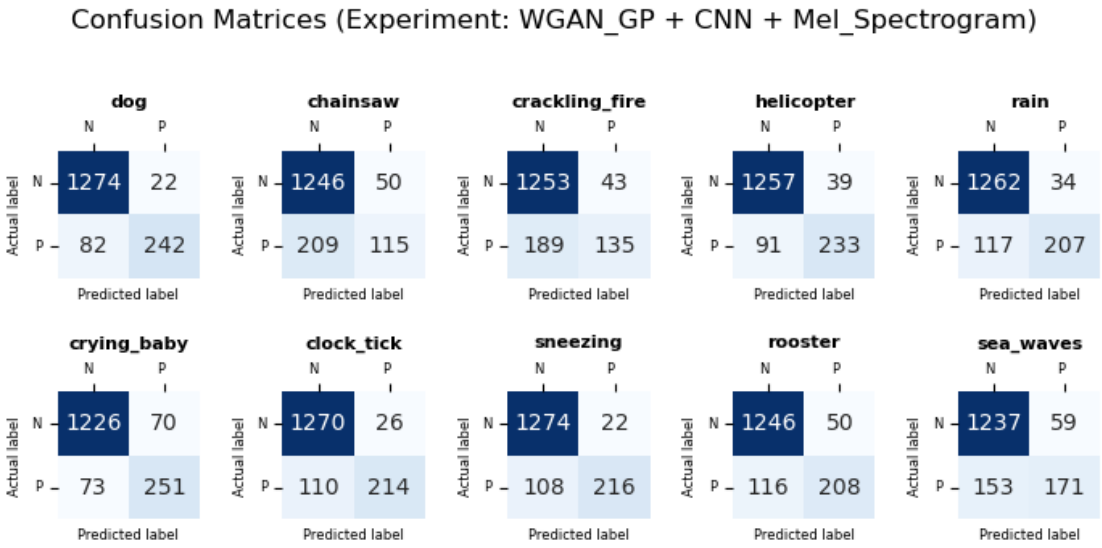


Figure 4.31: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	94.44	74.17	79.65
chainsaw	74.47	35.28	45.58
crackling_fire	73.71	43.06	53.2
helicopter	87.53	69.17	75.58
rain	89.34	63.06	72.29
crying_baby	80.57	76.39	77.68
clock_tick	89.27	65.83	73.55
sneezing	89.68	68.61	74.87
rooster	82.34	63.61	71.47
sea_waves	73.99	51.39	59.88
micro avg	82.14	61.06	70.02

Average Hamming Loss: 0.1
Average global accuracy (exact match): 45.72%
Average loss: 0.46

Figure 4.32: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed convolutional neural network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: Pitch Shifting, Model: CNN, Feature: Log-Mel Spectrograms

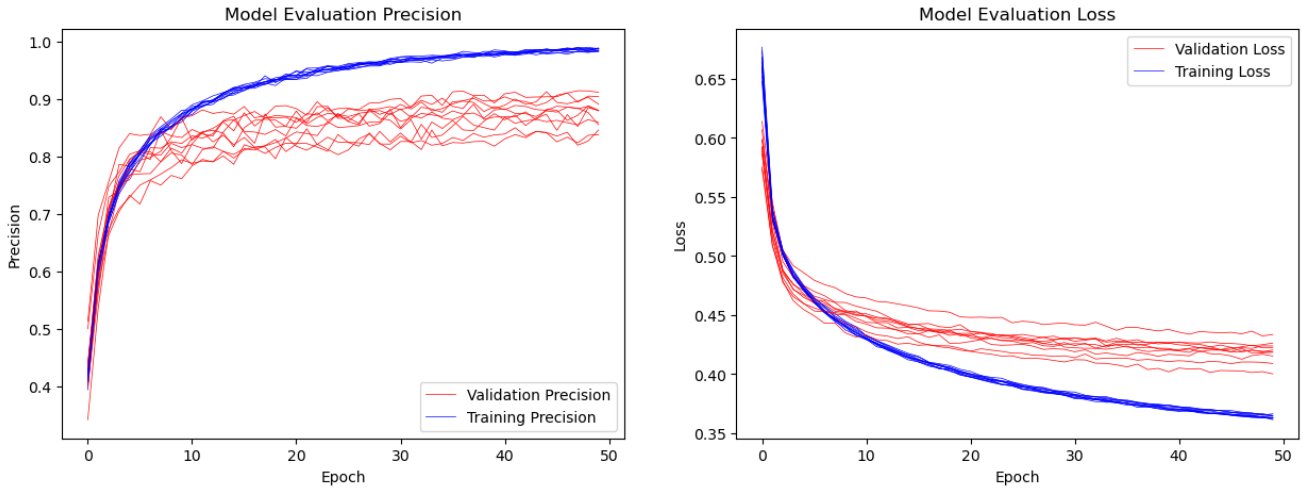


Figure 4.33: Precision and loss graphs obtained from evaluating the proposed convolutional neural network with a 10-fold cross validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the pitch shifting augmented files.

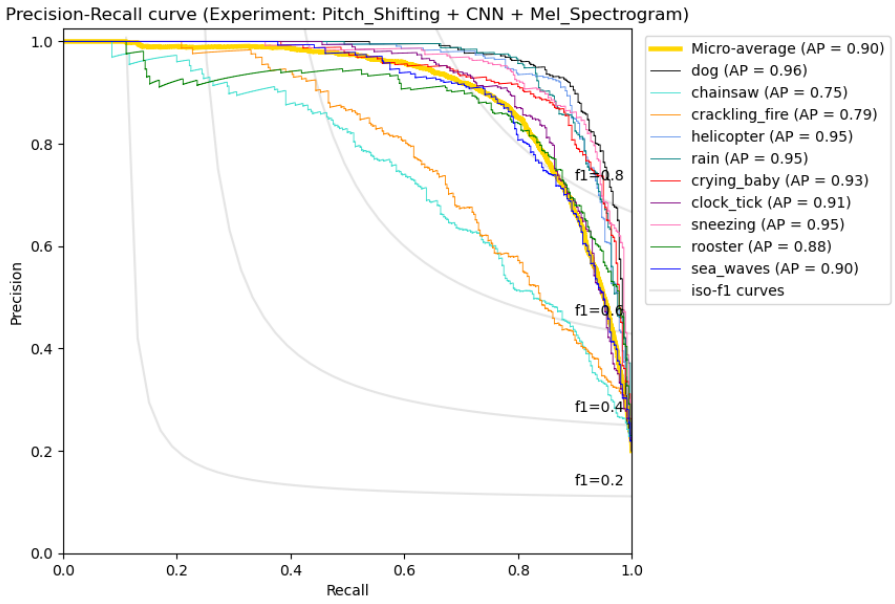


Figure 4.34: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.90 was obtained. Dog class obtained the highest value (AP = 0.96) while the worst was obtained by the chainsaw class (AP = 0.75).

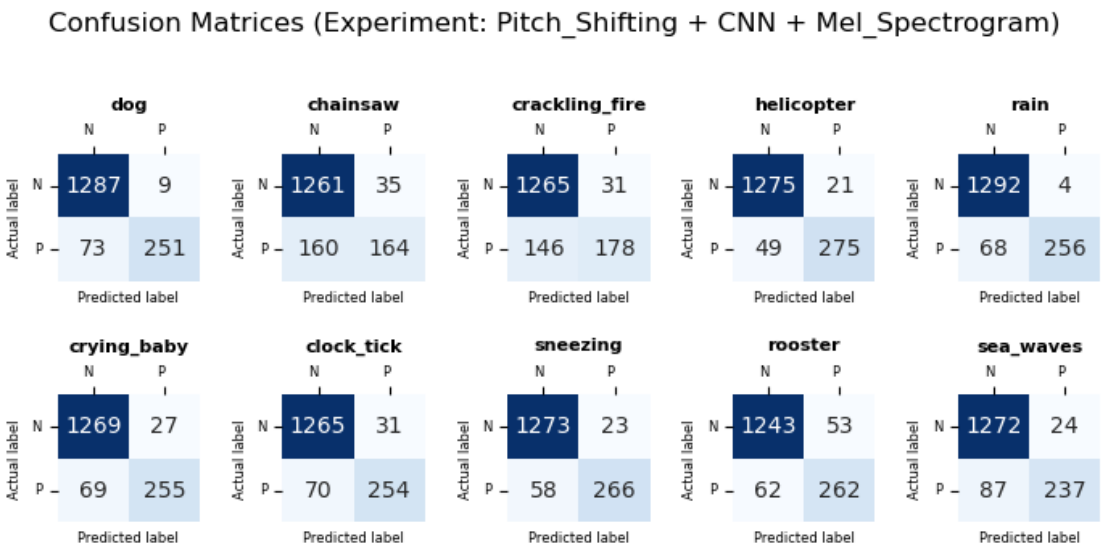


Figure 4.35: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	96.78	78.61	86.22
chainsaw	83.99	50.28	61.93
crackling_fire	84.23	53.89	65.37
helicopter	93.71	83.33	87.75
rain	97.06	79.72	87.06
crying_baby	90.94	80.83	84.86
clock_tick	89.97	79.17	82.31
sneezing	91.03	81.39	85.68
rooster	84.72	79.44	81.49
sea_waves	92.03	71.39	79.46
micro avg	90.19	73.81	81.15

Average Hamming Loss: 0.07
Average global accuracy (exact match): 44.94%
Average loss: 0.42

Figure 4.36: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed convolutional neural network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: N/A, Model: CNN, Feature: MFCCs

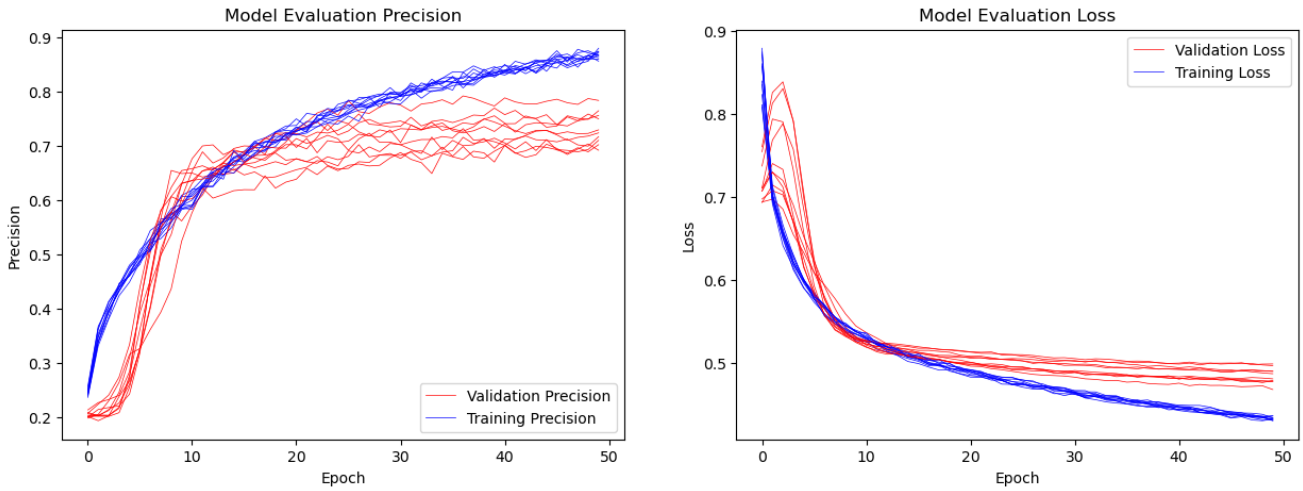


Figure 4.37: Precision and loss graphs obtained from evaluating the proposed convolutional neural network with a 10-fold cross validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset without data augmentation.

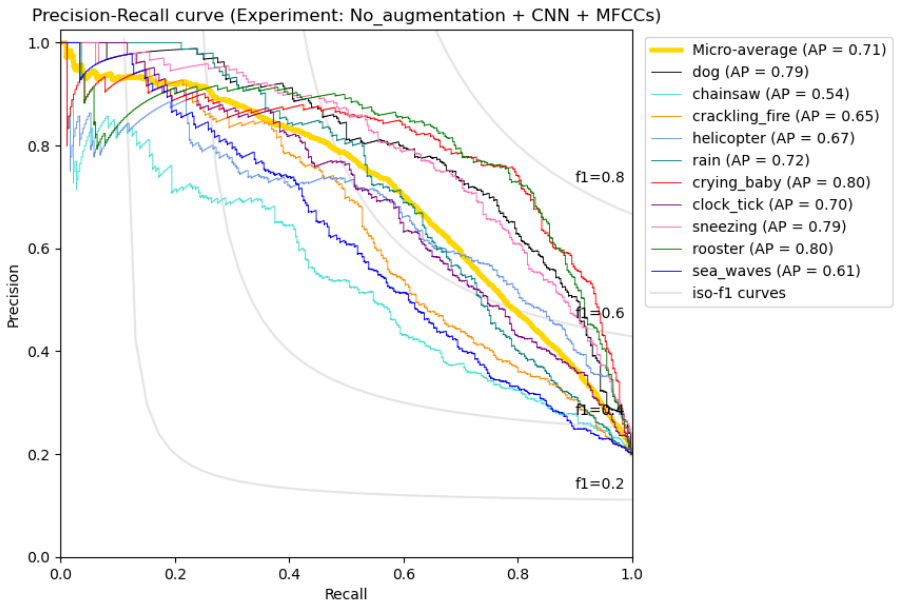


Figure 4.38: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.71 was obtained. Crying baby and rooster classes obtained the highest value (AP = 0.8) while the worst was obtained by the chainsaw class (AP = 0.54).

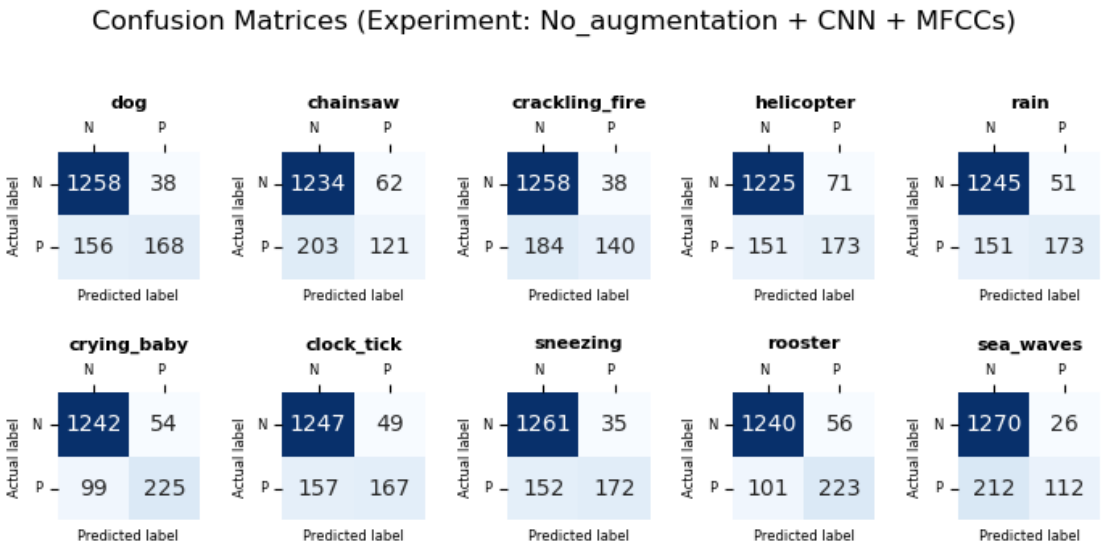


Figure 4.39: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	81.54	50.0	59.69
chainsaw	69.35	36.94	46.98
crackling_fire	78.68	41.94	53.9
helicopter	72.59	51.94	59.28
rain	79.2	53.33	62.58
crying_baby	80.46	72.5	74.91
clock_tick	76.69	47.5	54.71
sneezing	84.6	55.56	65.21
rooster	84.43	68.33	74.26
sea_waves	79.62	32.5	43.88
micro avg	77.41	51.06	61.47

Average Hamming Loss: 0.13
Average global accuracy (exact match): 36.89%
Average loss: 0.48

Figure 4.40: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed convolutional neural network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: WGAN-GP, Model: CNN, Feature: MFCCs

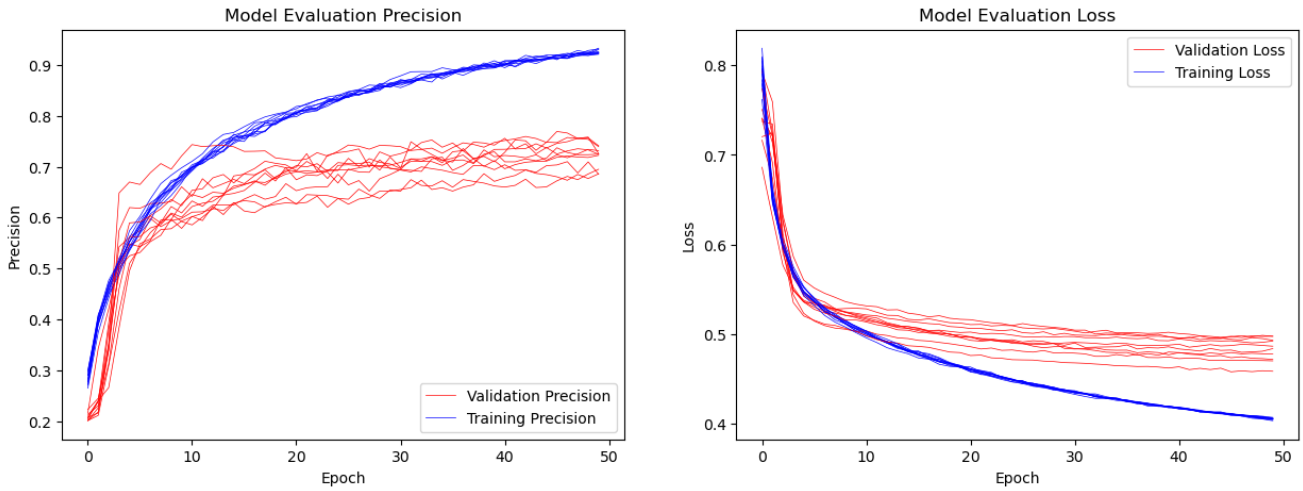


Figure 4.41: Precision and loss graphs obtained from evaluating the proposed convolutional neural network with a 10-fold cross validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the WGAN-GP augmented files.

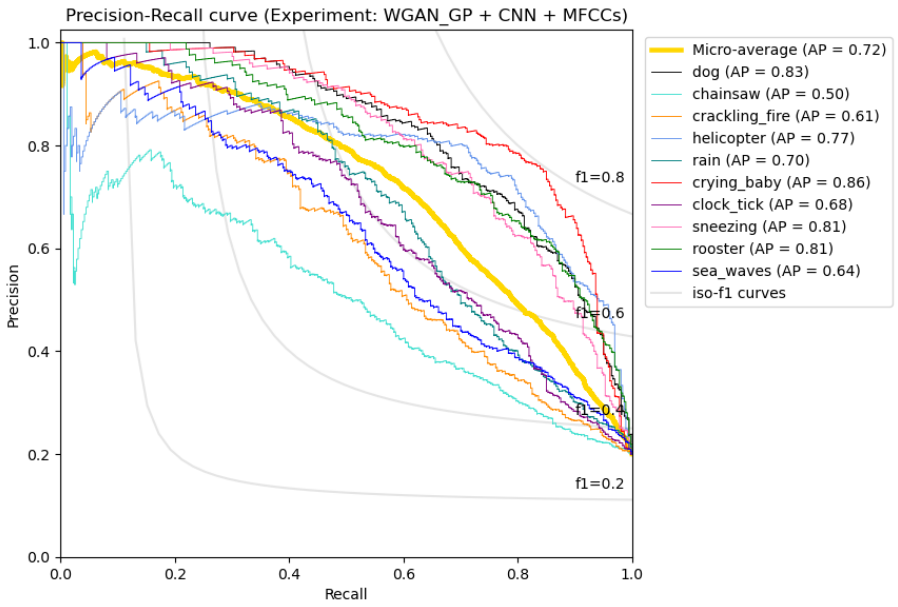


Figure 4.42: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.72 was obtained. Crying baby class obtained the highest value (AP = 0.86) while the worst was obtained by the chainsaw class (AP = 0.5).

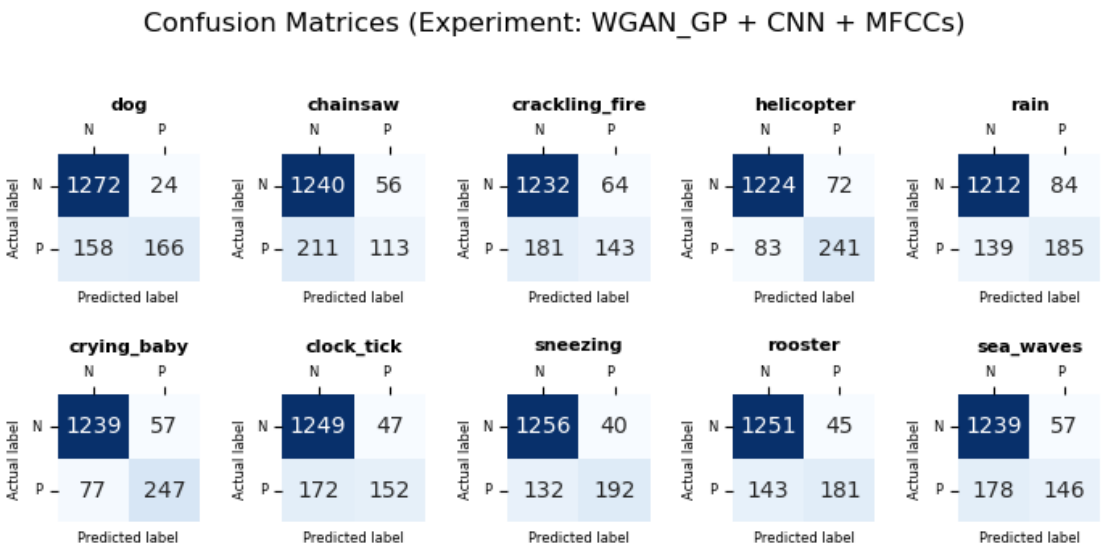


Figure 4.43: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	87.98	51.94	62.59
chainsaw	67.1	32.78	43.27
crackling_fire	69.66	45.0	53.57
helicopter	79.68	71.67	74.49
rain	74.23	57.5	61.97
crying_baby	83.04	75.56	78.2
clock_tick	78.36	47.22	55.36
sneezing	81.16	61.94	68.48
rooster	82.03	56.11	65.83
sea_waves	74.65	43.89	54.36
micro avg	76.32	54.36	63.45

Average Hamming Loss: 0.12
Average global accuracy (exact match): 38.67%
Average loss: 0.48

Figure 4.44: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed convolutional neural network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: Pitch Shifting, Model: CNN, Feature: MFCCs

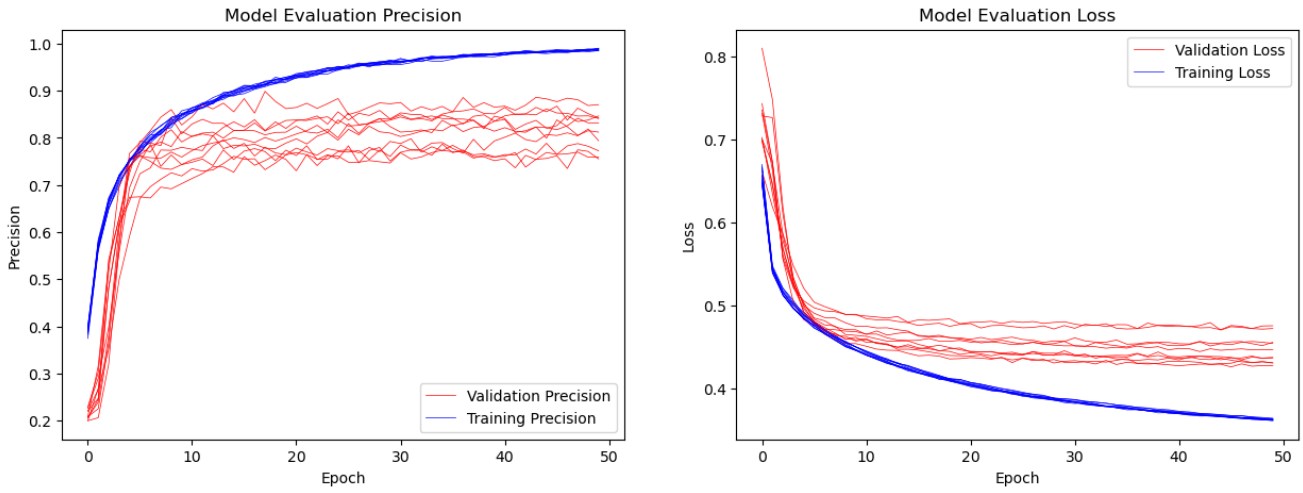


Figure 4.45: Precision and loss graphs obtained from evaluating the proposed convolutional neural network with a 10-fold cross validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the pitch shifting augmented files.

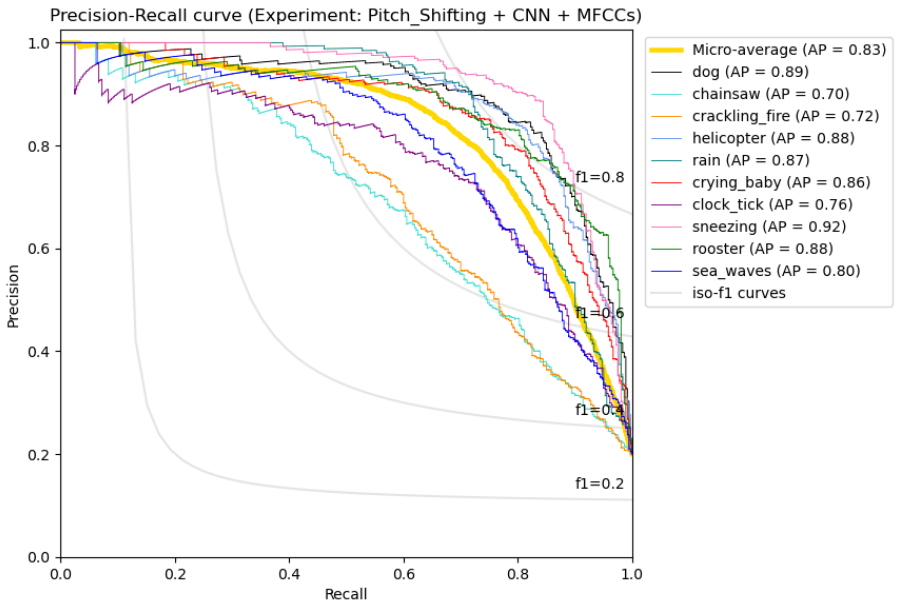


Figure 4.46: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.83 was obtained. Sneezing class obtained the highest value (AP = 0.92) while the worst was obtained by the chainsaw class (AP = 0.7).

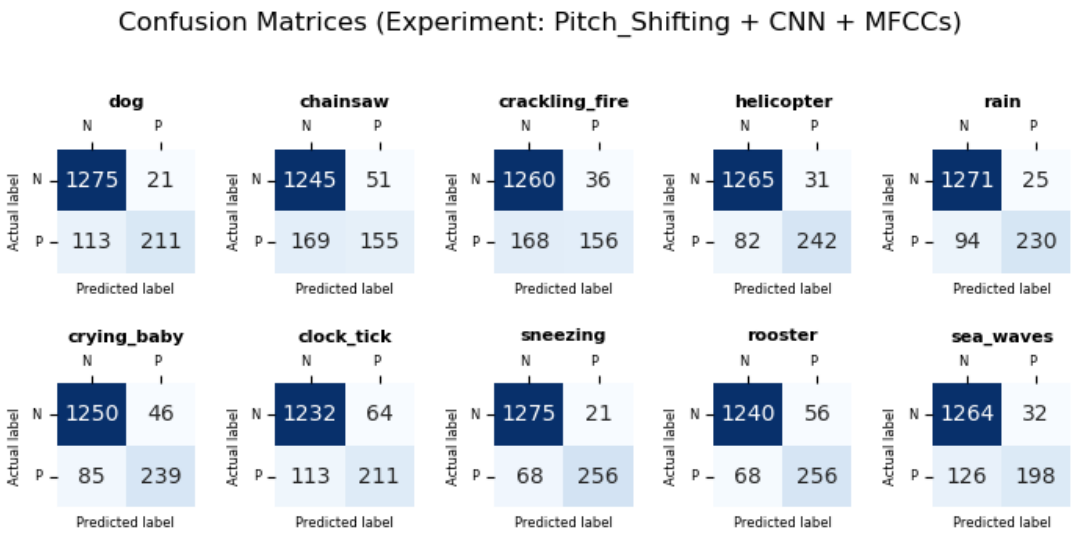


Figure 4.47: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.5.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	93.18	66.67	76.92
chainsaw	77.08	48.06	58.11
crackling_fire	83.63	48.33	60.56
helicopter	89.62	73.06	79.81
rain	89.69	72.22	78.87
crying_baby	84.15	75.83	78.81
clock_tick	77.55	64.44	68.92
sneezing	91.94	79.17	84.53
rooster	83.94	78.61	80.77
sea_waves	89.02	57.78	68.38
micro avg	84.84	66.42	74.47

Average Hamming Loss: 0.09
Average global accuracy (exact match): 42.72%
Average loss: 0.45

Figure 4.48: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed convolutional neural network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: N/A, Model: LSTM, Feature: Log-Mel Spectrograms

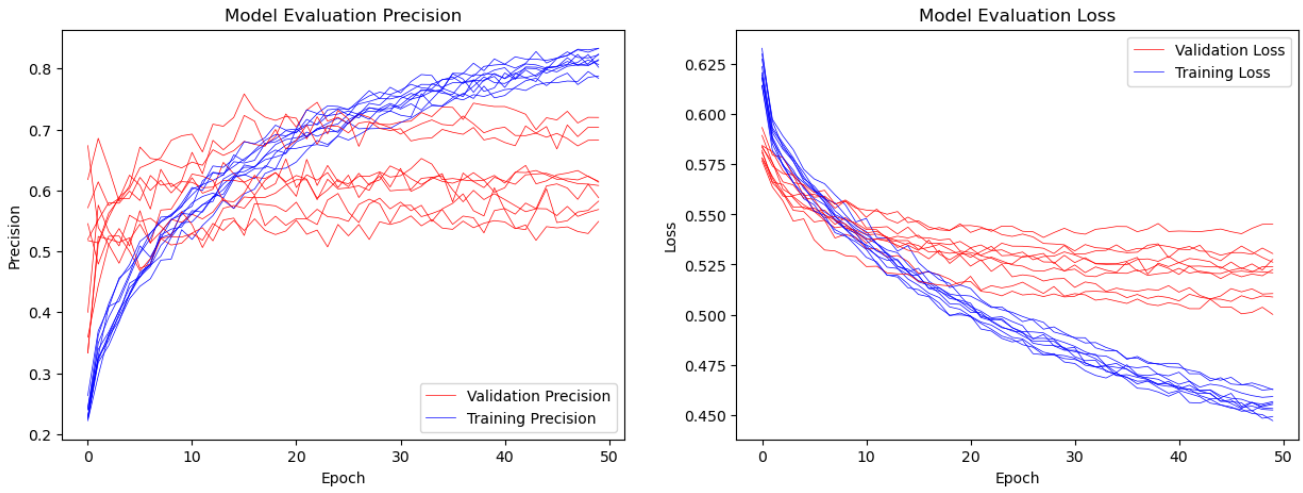


Figure 4.49: Precision and loss graphs obtained from evaluating the proposed LSTM network with a 10-fold cross-validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset without data augmentation.

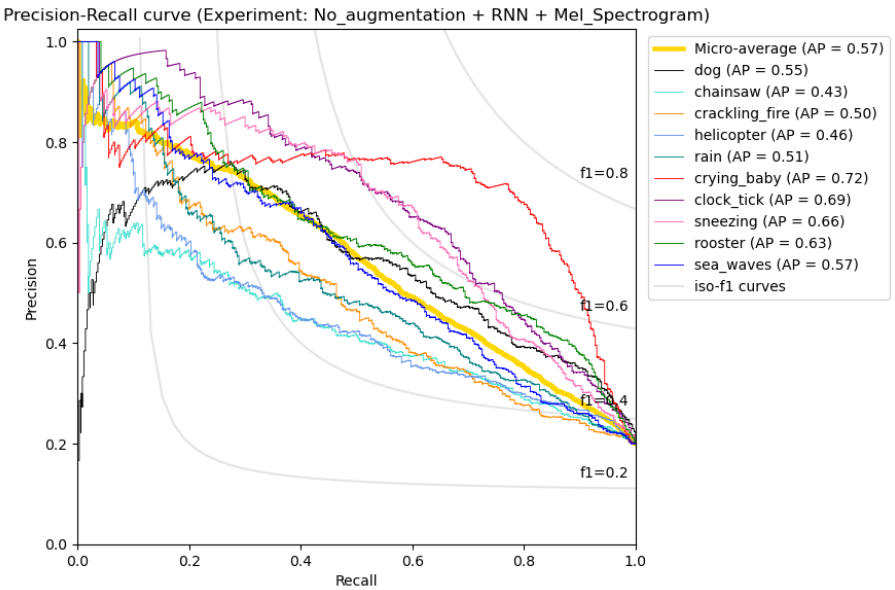


Figure 4.50: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.57 was obtained. Crying baby obtained the highest value (AP = 0.72) while the worst was obtained by the chainsaw class (AP = 0.43).

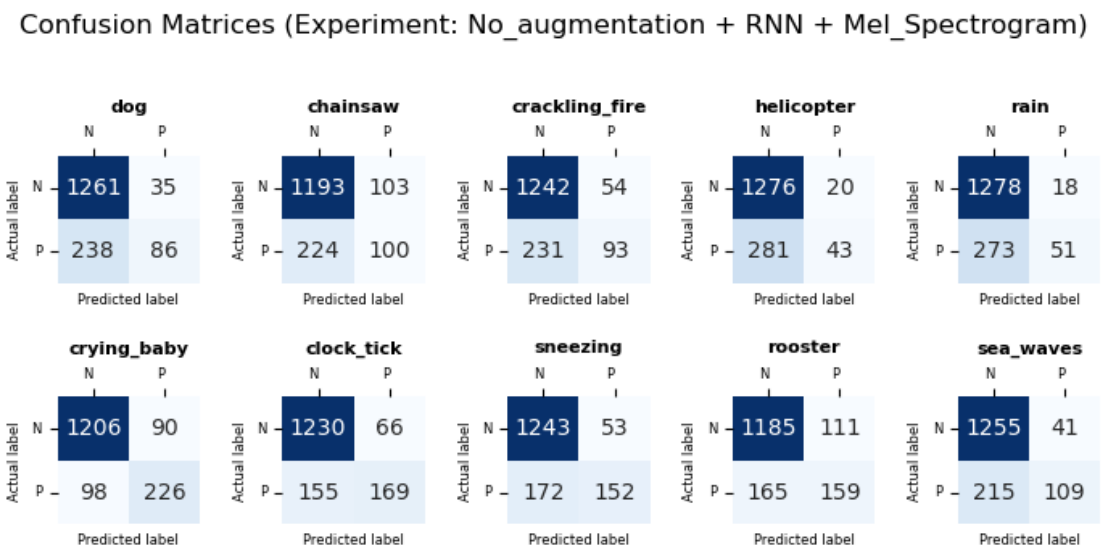


Figure 4.51: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	66.95	28.33	37.16
chainsaw	53.94	30.28	37.54
crackling_fire	63.53	27.5	36.52
helicopter	67.8	13.06	20.93
rain	69.01	15.83	25.11
crying_baby	72.59	72.5	71.45
clock_tick	73.03	51.94	58.17
sneezing	78.39	50.0	59.01
rooster	63.95	48.33	52.99
sea_waves	73.84	31.94	42.15
micro avg	67.66	36.97	47.79

Average Hamming Loss: 0.16
Average global accuracy (exact match): 27.94%
Average loss: 0.52

Figure 4.52: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed LSTM network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: WGAN-GP, Model: LSTM, Feature: Log-Mel Spectrograms

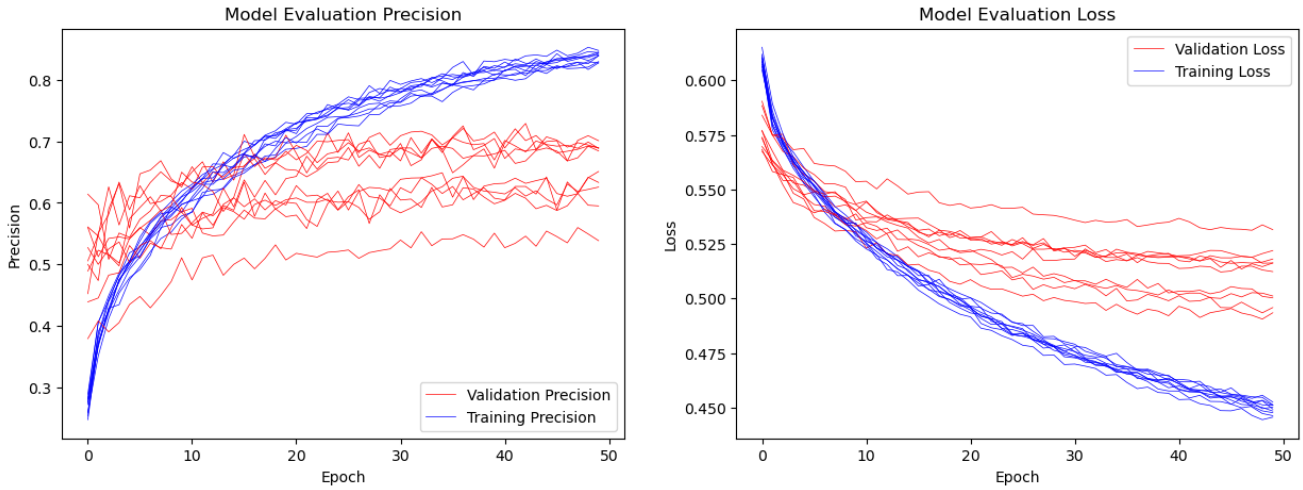


Figure 4.53: Precision and loss graphs obtained from evaluating the proposed LSTM network with a 10-fold cross-validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the WGAN-GP augmented files.

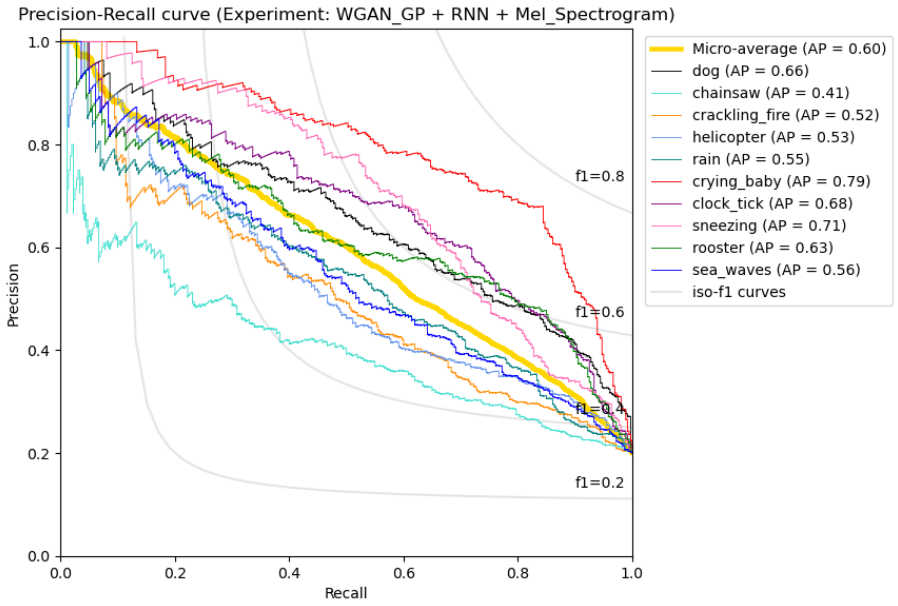


Figure 4.54: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.6 was obtained. Crying baby obtained the highest value (AP = 0.79) while the worst was obtained by the chainsaw class (AP = 0.41).

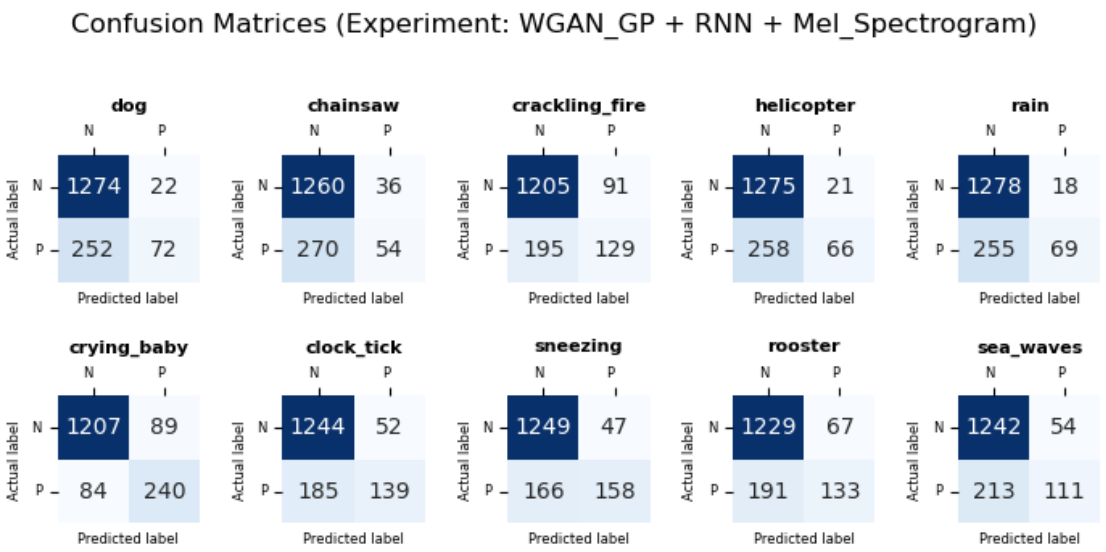


Figure 4.55: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	71.89	23.89	34.68
chainsaw	61.49	16.67	25.14
crackling_fire	58.86	39.72	46.07
helicopter	62.48	18.33	27.13
rain	78.01	21.67	32.18
crying_baby	75.78	72.5	73.1
clock_tick	76.59	41.94	51.62
sneezing	77.68	51.11	60.51
rooster	67.55	39.44	47.56
sea_waves	66.81	32.5	42.46
micro avg	69.57	35.78	47.21

Average Hamming Loss: 0.16
Average global accuracy (exact match): 27.56%
Average loss: 0.51

Figure 4.56: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed LSTM network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: Pitch Shifting, Model: LSTM, Feature: Log-Mel Spectrograms

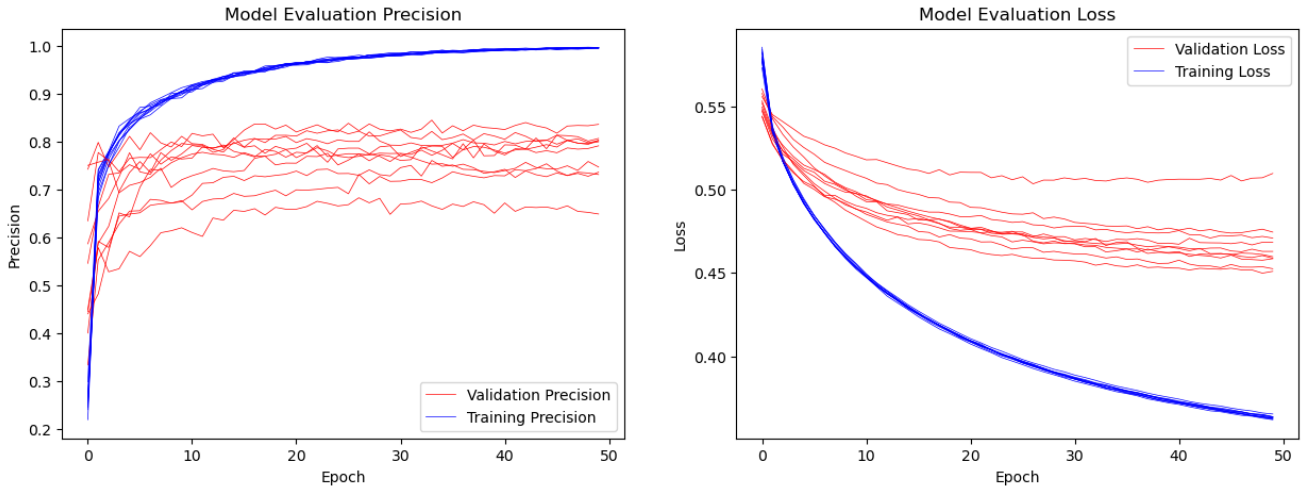


Figure 4.57: Precision and loss graphs obtained from evaluating the proposed LSTM network with a 10-fold cross-validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the pitch shifting augmented files.

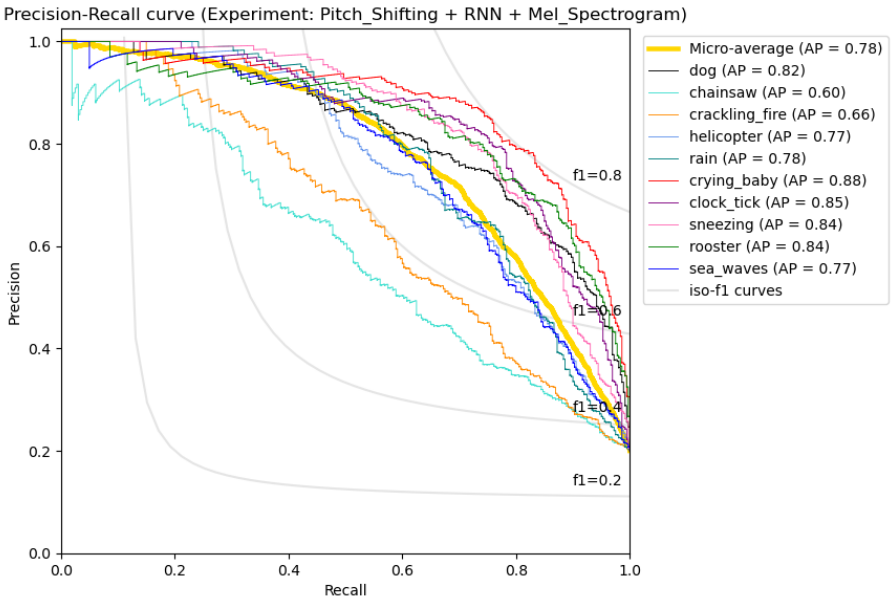


Figure 4.58: Precision-recall curves of each class for the multi-label classification task. An average precision of 0.78 was obtained. Crying baby class obtained the highest value (AP = 0.88) while the worst was obtained by the chainsaw class (AP = 0.6).

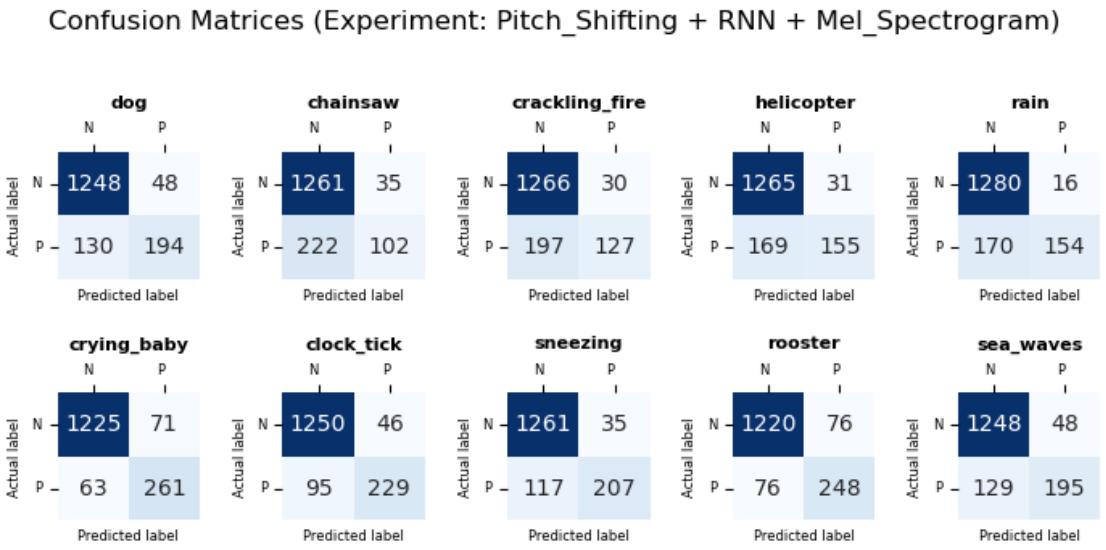


Figure 4.59: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	81.74	61.39	69.47
chainsaw	75.02	32.22	44.17
crackling_fire	78.7	38.61	51.5
helicopter	84.31	48.89	61.29
rain	89.96	49.72	62.78
crying_baby	79.72	82.22	80.15
clock_tick	86.32	70.83	75.84
sneezing	86.04	64.72	73.19
rooster	80.05	75.0	76.68
sea_waves	81.71	58.06	66.83
micro avg	81.47	58.17	67.81

Average Hamming Loss: 0.11
Average global accuracy (exact match): 36.11%
Average loss: 0.47

Figure 4.60: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed LSTM network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: N/A, Model: LSTM, Feature: MFCCs

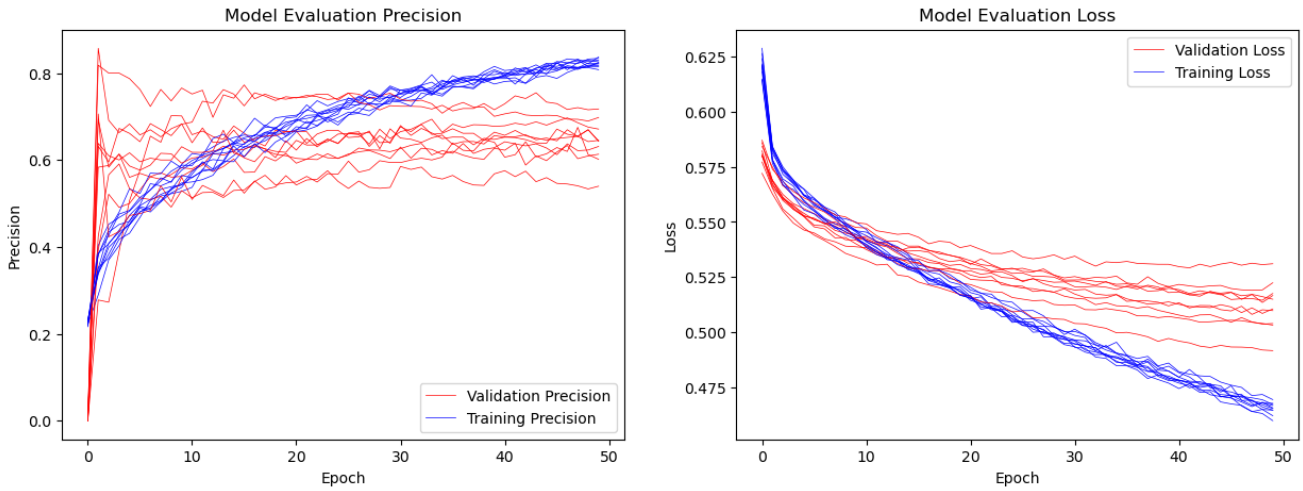


Figure 4.61: Precision and loss graphs obtained from evaluating the proposed LSTM network with a 10-fold cross-validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset without data augmentation.

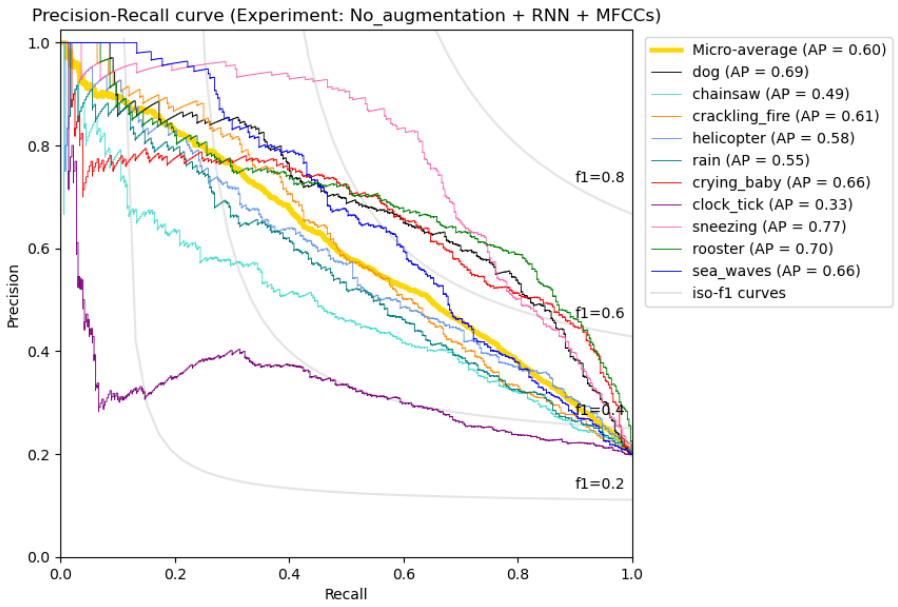


Figure 4.62: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.6 was obtained. Sneezing obtained the highest value (AP = 0.77) while the worst was obtained by the clock tick class (AP = 0.33).

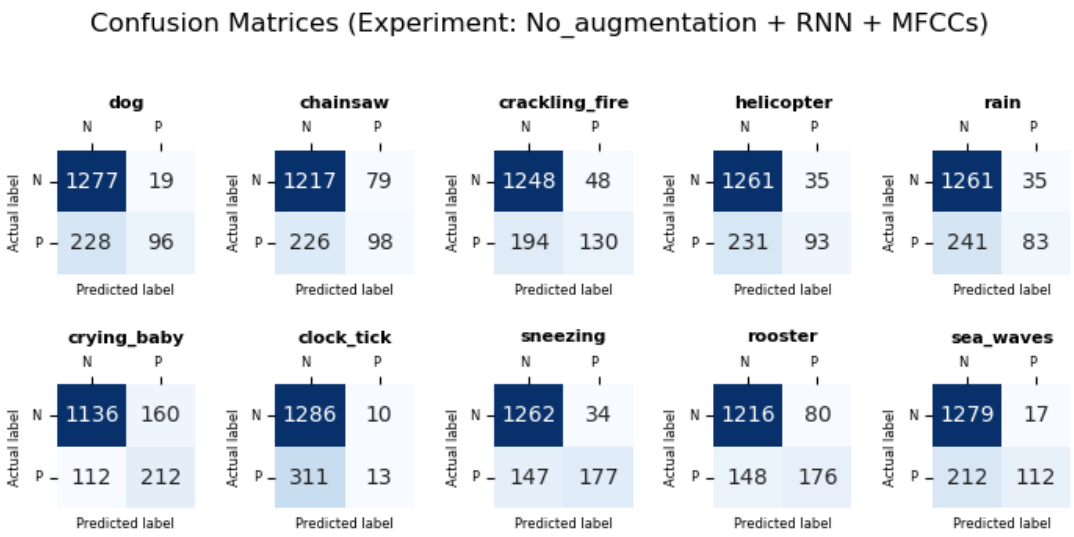


Figure 4.63: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	83.21	31.11	43.39
chainsaw	62.32	30.28	39.38
crackling_fire	72.74	38.89	49.81
helicopter	80.2	26.11	36.35
rain	73.89	26.11	37.91
crying_baby	57.4	68.89	61.36
clock_tick	36.72	4.17	7.17
sneezing	86.59	57.78	66.86
rooster	76.67	52.5	59.83
sea_waves	82.15	31.39	43.57
micro avg	70.13	36.72	48.14

Average Hamming Loss: 0.16
Average global accuracy (exact match): 28.89%
Average loss: 0.51

Figure 4.64: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed LSTM network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: WGAN-GP, Model: LSTM, Feature: MFCCs

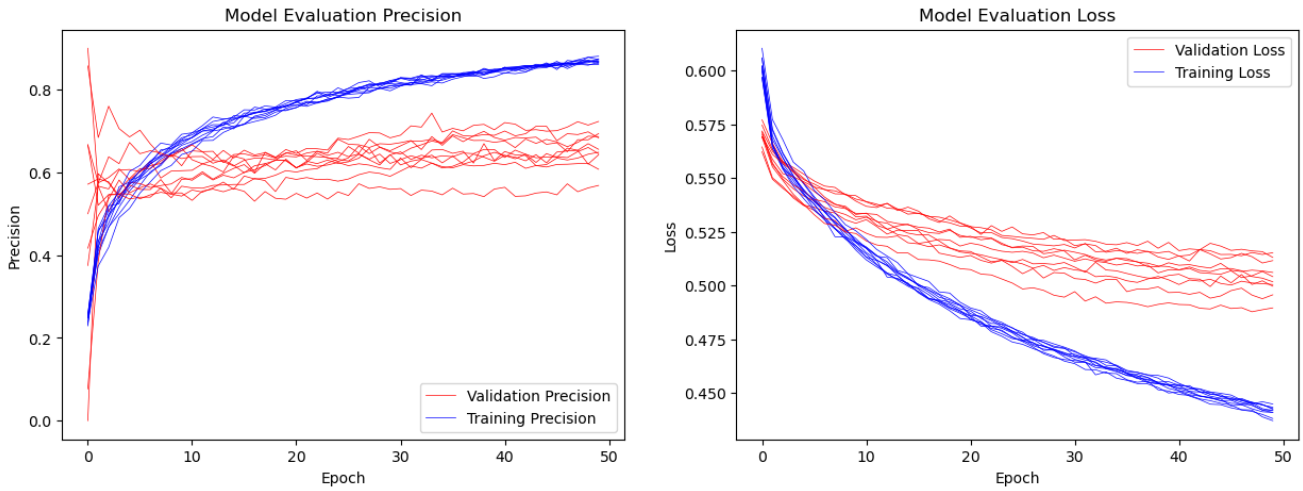


Figure 4.65: Precision and loss graphs obtained from evaluating the proposed LSTM network with a 10-fold cross-validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the WGAN-GP augmented files.

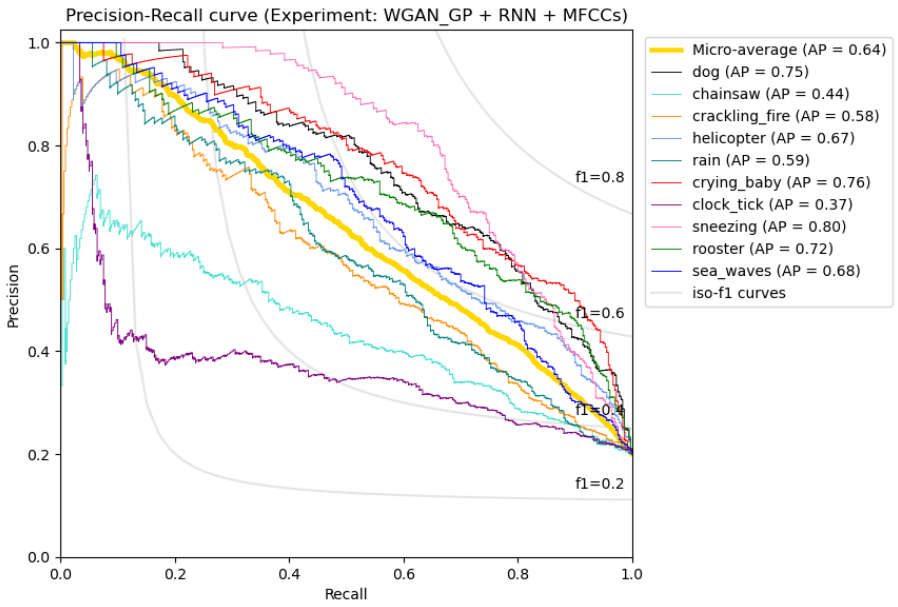


Figure 4.66: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.64 was obtained. Sneezing obtained the highest value (AP = 0.8) while the worst was obtained by the clock tick class (AP = 0.37).

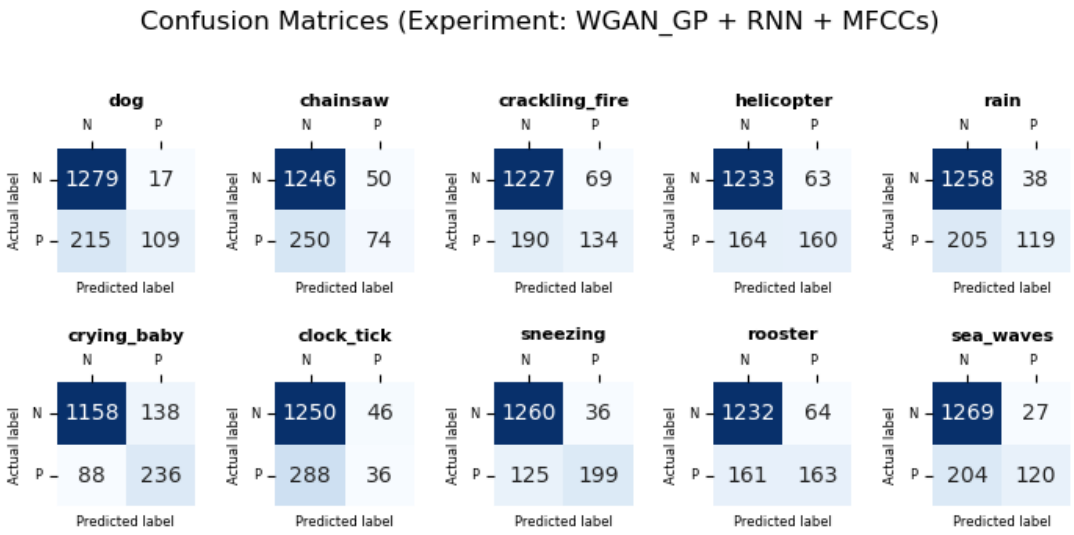


Figure 4.67: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	79.64	34.17	45.0
chainsaw	69.34	22.78	32.15
crackling_fire	64.48	41.94	50.22
helicopter	77.78	46.39	54.7
rain	80.74	36.11	46.28
crying_baby	65.55	72.5	68.12
clock_tick	35.57	10.56	15.48
sneezing	83.51	64.17	70.37
rooster	77.18	49.72	58.85
sea_waves	81.74	36.39	48.01
micro avg	70.45	41.47	52.15

Average Hamming Loss: 0.15
Average global accuracy (exact match): 31.67%
Average loss: 0.5

Figure 4.68: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed LSTM network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Data Augmentation: Pitch Shifting, Model: LSTM, Feature: MFCCs

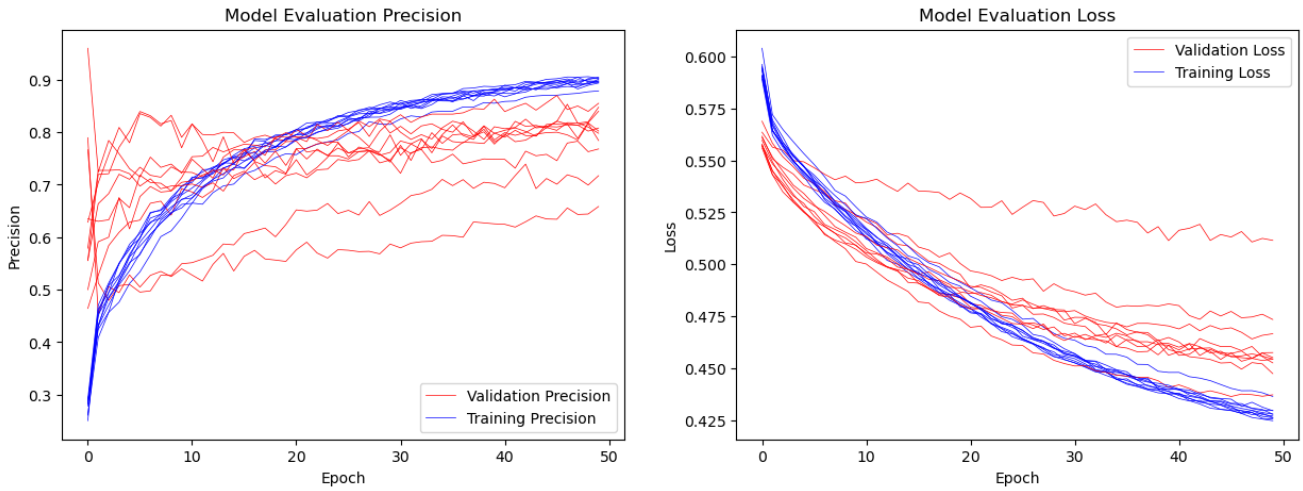


Figure 4.69: Precision and loss graphs obtained from evaluating the proposed LSTM network with a 10-fold cross-validation, using as input log-mel spectrograms extracted from processing the ESC-10 dataset and the pitch shifting augmented files.

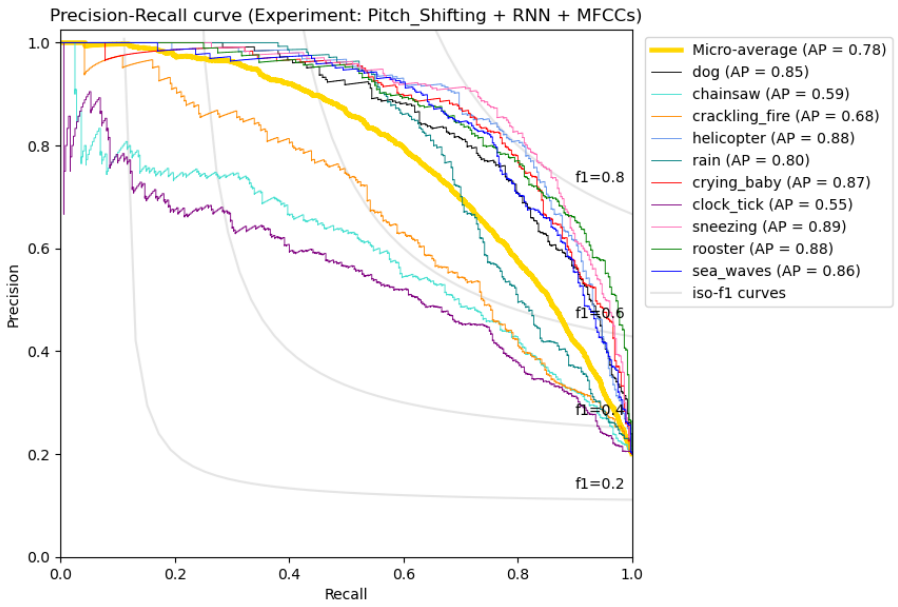


Figure 4.70: Precision-recall curves of each class for the multi-label classification task. An average precision (area under the curve) of 0.78 was obtained. Sneezing obtained the highest value (AP = 0.89) while the worst was obtained by the clock tick class (AP = 0.55).

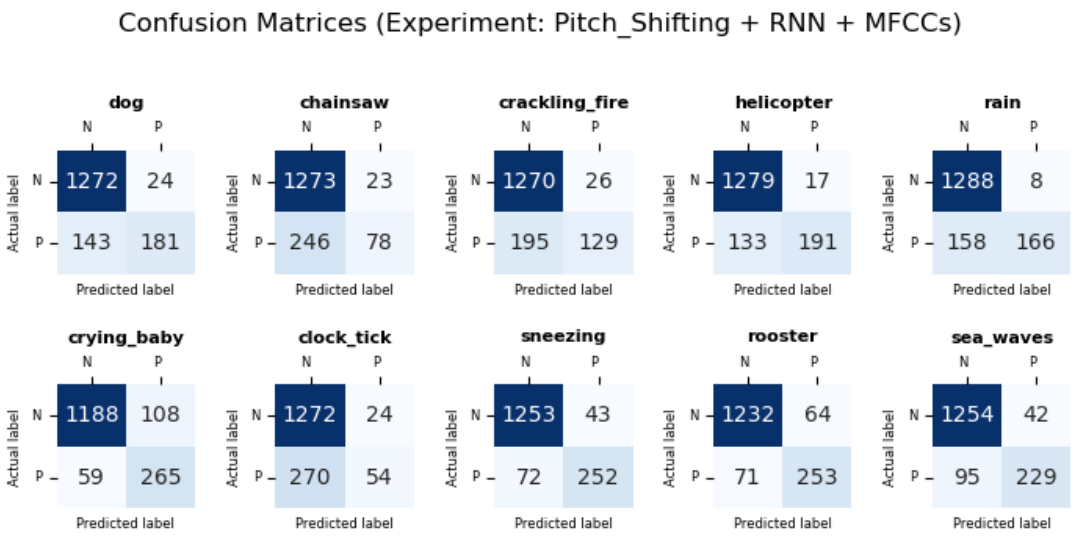


Figure 4.71: Confusion matrices of each class obtained from evaluating the proposed model using a threshold value of 0.55.

K-fold cross validation averages

	Precision (%)	Recall (%)	F1 (%)
dog	88.64	57.78	68.98
chainsaw	80.93	23.89	35.61
crackling_fire	79.77	38.06	51.02
helicopter	92.71	60.56	72.01
rain	95.63	52.78	66.39
crying_baby	72.52	83.61	76.91
clock_tick	72.32	18.06	27.31
sneezing	85.78	78.06	81.33
rooster	82.43	76.11	78.42
sea_waves	86.99	67.78	75.17
micro avg	82.61	55.67	66.48

Average Hamming Loss: 0.11
Average global accuracy (exact match): 33.0%
Average loss: 0.46

Figure 4.72: Precision, recall and F1-score values for each of the ten classes from the ESC-10 dataset. These values where obtained by evaluating the proposed LSTM network with a 10-fold cross-validation using a threshold value of 0.55. The micro-average of these metrics, Hamming Loss, global accuracy and training loss values are also shown.

Neural Network + Feature	**Data Aug.**	**No. Files**	**H. Loss**	**Prec.**	**Recall**	**F1 Score**	**Global Acc.**	**Tr. Loss**	**AP**
CNN + Log-Mel Spectro-grams lr = 0.00001	N\A	1800	0.12	81.14%	55.25%	65.7%	41.11%	0.47	0.76
	WGAN-GP	3600	0.1	82.14%	61.06%	70.02%	**45.72%**	0.46	0.79
	Time Stretching	3600	0.09	86.18%	64.14%	73.5%	44.33%	0.44	0.84
	Pitch Shifting	3600	**0.07**	**90.19%**	**73.81%**	**81.15%**	44.94%	**0.42**	**0.90**
CNN + MFCCs lr = 0.000025	N\A	1800	0.13	77.41%	51.06%	61.47%	36.89%	0.48	0.71
	WGAN-GP	3600	0.12	76.32%	54.36%	63.45%	38.67%	0.48	0.72
	Time Stretching	3600	0.11	82.69%	57.06%	67.48%	40.61%	0.46	0.79
	Pitch Shifting	3600	**0.09**	**84.84%**	**66.42%**	**74.47%**	**42.72%**	**0.45**	**0.83**
LSTM + Log-Mel Spectro-gram lr = 0.00001	N\A	1800	0.16	67.66%	36.97%	47.79%	27.94%	0.52	0.57
	WGAN-GP	3600	0.16	69.57%	35.78%	47.21%	27.56%	0.51	0.6
	Time Stretching	3600	0.12	**83.02%**	50.97%	63.13%	33.39%	**0.47**	0.76
	Pitch Shifting	3600	**0.11**	81.47%	**58.17%**	**67.81%**	**36.11%**	**0.47**	**0.78**
LSTM + MFCCs lr = 0.000025	N\A	1800	0.16	70.13%	36.72%	48.14%	28.89%	0.51	0.6
	WGAN-GP	3600	0.15	70.45%	41.47%	52.15%	31.67%	0.5	0.64
	Time Stretching	3600	0.14	75.39%	46.64%	57.58%	32.28%	0.49	0.68
	Pitch Shifting	3600	**0.11**	**82.61%**	**55.67%**	**66.48%**	**33%**	**0.46**	**0.78**

Table 4.2: Results of the polyphonic stage. The best results of each configuration (Neural Network + feature) are highlighted in bold.

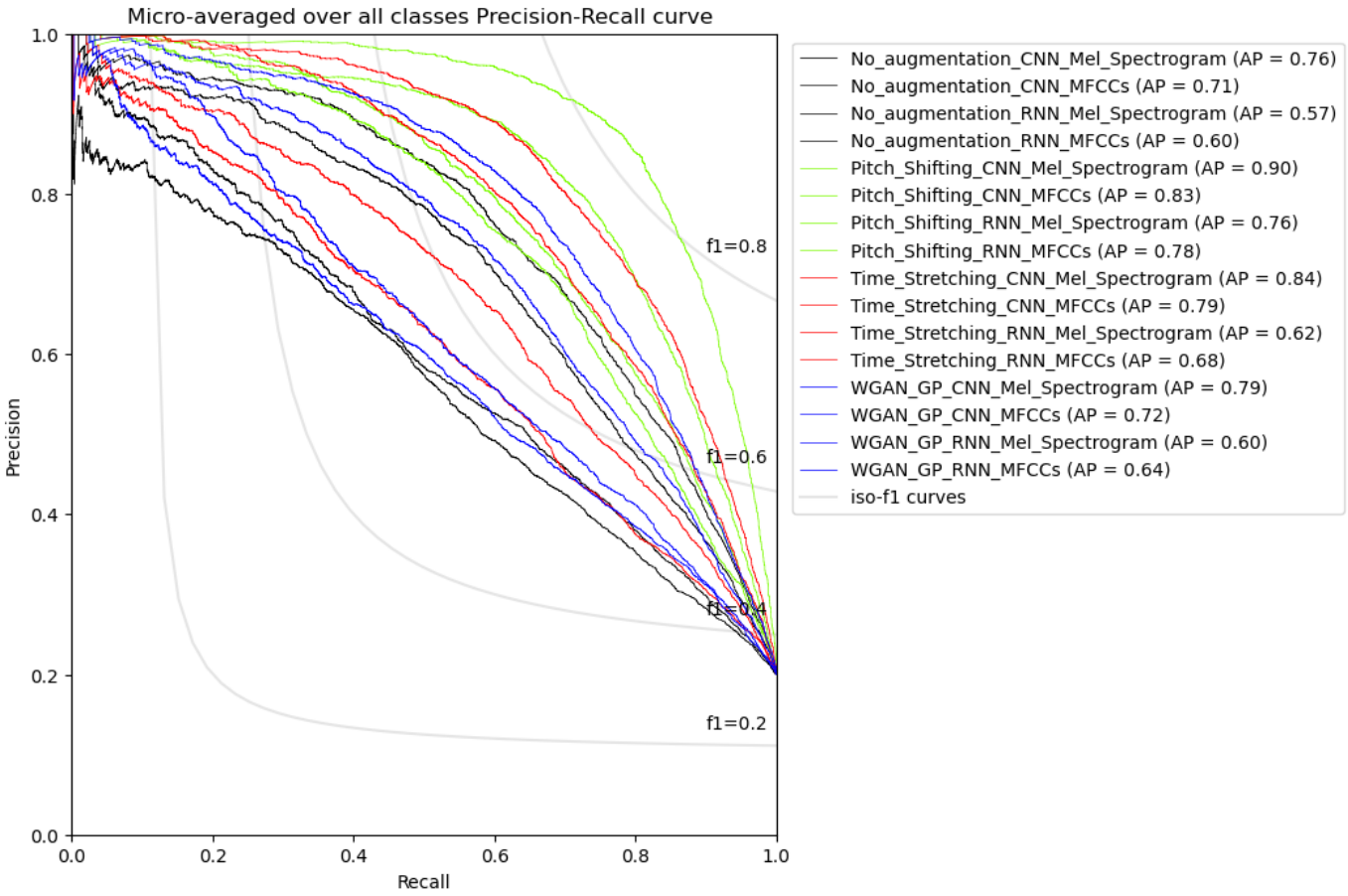


Figure 4.73: Micro-averaged precision-recall curves over all classes from all the experiments made. AP is the average precision which is equivalent to the area under the curve. The blue lines belong to the models which use the WGAN-GP as augmentation method, the green lines the pitch shifting method, the red lines the time stretching method and the black lines the polyphonic dataset without augmentation.

4.2 Discussion

In general the three data augmentation methods that were tested improve the overall performance of the two different neural networks, being pitch shifting the best method, followed by time stretching and the WGAN-GP in third place. This is reflected on the evaluation metrics that were computed, summarized in Table 3.1 and Table 3.2.

For the environmental sound classification, i.e., the monophonic classification task, the higher scores were obtained by the convolutional neural network using log-mel spectrograms as features and pitch shifting as data augmentation method, obtaining 92% in accuracy, which is an improve of 11.76% with respect to the baseline (80.24%), 5.25% with respect to the WGAN-GP augmentation method (86.75%) and 3.75% with respect to the time stretching method (88.25%). The other metrics yielded values similar to those obtained for accuracy, which confirms the robustness of the proposed models. About the LSTM network, interestingly, the pitch shifting augmented dataset surpassed to a greater extent the rest of the datasets with both MFCCs (80%) and log-mel spectrograms (86.25%) in accuracy, indicating this data augmentation method introduces higher inter-class and also intra-class variations, which enhances the learning ability of the neural network. On the other hand, LSTM's best score with the WGAN-GP augmented dataset is 71.25% in accuracy.

Comparing these results to the ones obtained in other researches, it is clear that the methodology proposed is adequate to resolve the task. A relevant work to be taken as initial reference, since the author created this dataset and is the first that used a CNN for this task, obtained 80% in accuracy [2]. In other article, the authors used a different type of generative adversarial network for data augmentation with a CNN as classifier. They scored 96% in accuracy for the same dataset [6]. They also compared the performance of this method against some classical augmentation methods, obtaining higher results with the generative deep model. This fact reaffirms the thought that further investigation has to be done with the model of generative adversarial network implemented in this work. Another work used a support vector machine (SVM) and a random forest as classifiers, plus a different type of GAN called WCCGAN that generates spectrogram images instead of raw audio [17]. They achieved 87% in accuracy using this method of data augmentation, slightly above the value obtained using the WGAN-GP and below the best model of this work which used CNN and pitch shifting as augmentation technique. They also obtained 71% without data augmentation, which compared to the baseline of this work is considerably low. In a different work

[27], the authors compared a CNN and a LSTM network for the UrbanSound8K dataset and different features which include MFCCs and Mel Spectrograms. Their LSTM network obtained higher accuracy values with most of the features tested over the ones obtained by the CNN, being the higher 98.23% using data augmentation. This may indicate that LSTM networks are more sensitive to dataset size, since in this work it was observed that this network overfitted faster than the CNN.

For the Audio Tagging task, i.e., the polyphonic approach, the tendency was similar, however, in this case the main metrics used as indicators of the robustness of the different models were Hamming loss (HL) and F1-Score, the first for representing an appropriate substitute for accuracy and the second for being the harmonic mean of precision and recall. The higher scores were obtained using the CNN, being the best model the one that use the pitch shifting augmented dataset and log-mel spectrograms (HL = 0.07, F1 = 81.15%), followed by the pitch shifting dataset with MFCCs (HL = 0.09, F1 = 74.47%), the time stretching dataset with log-mel spectrograms (HL = 0.09, F1 = 73.5%) and the WGAN-GP method (HL = 0.1, F1 = 70.02%). The best model of the baseline (no augmentation) reached a Hamming loss of 0.12 and a F1-score of 65.7%. On the other hand, the LSTM network that achieved higher values used pitch shifting as augmentation method and mel-spectrograms as input (HL = 0.11, F1 = 67.81%). Compared to the CNN, in these experiments it was observed a more pronounced difference between (high) precision and (low) recall, suggesting LSTMs tend to be more selective in its predictions, i.e., most of its predictions are correct, however, it is hesitant to label a class as positive (present) in an audio, probably due to less ability to generalize. This would also explain also why the Label Smooth method had a positive impact on the overall performance of this network.

From the micro-averaged over all classes precision-recall curves from Figure 4.73 and their respective average precision values (AP), it can be observed the hierarchy of the different models tested for this task, being the more robust, clearly, those that use the CNN as classifier. Furthermore, the analysis of the precision-recall curves for each class (from each model) also provides interesting information about the learning capacity of proposed models. Remembering that a multi-label task with n classes can be decomposed into n binary classification problems, we can see through these graphs, specifically through its AP value, for which classes a model is not learning. Those curves that have an average precision slightly above or below 50% represent classes that the model is not learning to classify correctly. The model which is represented by Figure 4.50 is a case in which the classifier is clearly

not learning to distinguish most but a couple of classes. This implies that by varying the threshold (set to 0.55 for this work), for an increase in precision, there would be a decrease in recall, and vice versa. The opposite example can be observed in Figure 4.34, in which it is observed that all the curves have high AP values, which means that varying the threshold will not significantly modify the precision and recall values, which is the behaviour that would be expected from a robust model.

Before running the experiments it was expected in advance that the classifiers would not show the same learning ability over all the classes, classifying better some than others. This behaviour was clearly observed on each model through the precision-recall curves, the confusion matrices and the evaluation metrics. The class with which the classifiers had the most trouble, both for monophonic and polyphonic classification tasks, was the chainsaw. This may have to do with the inner nature of this type of sound, since it posses a higher content of high frequencies with high amplitudes, thus more acoustic energy concentrated in that frequency range (see Figure 3.6). There were experiments where data augmentation apparently didn't showed a positive effect over the learning ability of the classifiers, however, when analysed case by case it was observed the reason had to do with specific classes getting affected by augmentation. An example can be found on the environmental sound classification task (monophonic classification) using the CNN as classifier and MFCCs as features; seeing the confusion matrices from Figures 4.8, 4.10 and 4.12 it can be seen the chainsaw class exhibited lower performance when using the augmented datasets. Even pitch shifting, the best method, couldn't enhance the performance of the CNN on that specific class. Other classes that presented low metrics for some models were the clock and crackling fire, specially for the Audio Tagging task.

One of the aspects that makes this work original is the use of a generative adversarial network as data augmentation method to address environmental audio tagging, since no other work was found that used this technique for this specific task. Other techniques have been implemented, though. Another aspect has to do with the construction of a polyphonic database from a monophonic one, since there is very little literature where a similar methodology is carried out and apparently no work that does it with the ESC-10 dataset, although, other datasets have been used to study this task. Some examples are mentioned below.

In one paper, the authors synthesized new samples using an augmentation method called mixup and trained a convolutional recurrent neural network (CRNN)[28]. Their experiments were conducted on the DCASE 2016 Task 4 dataset, focused in domestic environmental

sounds. Their results, based on the equal error rate metric, showed their proposed method of data augmentation can effectively improve the performance of audio tagging. A different approach was followed in other research, where the authors used pre-trained models, specifically various types of EfficientNet, a MobileNet V2 and a ResNet-50, all incorporating attention modules. They tested their different models on a dataset called AudioSet, which is "a collection of over 2 million 10-second audio clips excised from YouTube videos and labeled with the sounds that the clip contains from a set of 527 labels", for which they scored 47.44% as their higher value using mAP as their main evaluation metric. Another work obtained a Hamming Loss of 0.03 and F1-score 62.8% (presented as accuracy) using a Multi Label DNN on a self-made dataset (61 classes of sounds) composed of audio recordings collected from "highly realistic everyday environments" [29]. An example with a very specific application was presented in a research, where the authors used a self-made dataset containing audio-clips (1435 samples) recorded in natural environments from which they identified five main classes of sounds, birds, insects, low activity, rain and wind. They did multi-label classification on this dataset, obtaining a Hamming Loss of 0.07 with their best model, which was a Multilayer Perceptron [30].

Researches where the authors build their polyphonic dataset from a monophonic one are the following. In the first work, the UrbanSound8K dataset was used to create a polyphonic dataset. Their best model achieved 65% in prediction accuracy for the single-label classification task and tor the Audio Tagging task they scored 70.56% in precision and 58% in recall. Other datasets have been used to create polyphonic datasets, for example, in [31] the authors mixed two datasets that contained monophonic sounds from two different videogames. For the classification they tested one convolutional recurrent neural network (CRNN) and a novel method called RARE (Real-time Audio Recognition Engine) and they considered as evaluation metrics F1-score and the error rate. Their results showed the second method outperformed the first one, obtaining an f1-score of 74.5% while the CRNN obtained 56.4%.

About using the WGAN-GP network as augmentation method, as it has already been mentioned, it obtained lower scores compared to the other two augmentation techniques. The reason, it seems, has to do with the quality of the sounds synthesized by the generator for some of the classes, since when comparing models where only the dataset was varied, it was possible to observe that for the WGAN-GP some classes showed an increase in the evaluation scores, but not as good as the increase shown by using the other techniques; in some cases even worsening the values with respect to the baseline. Also, the training-validation

and loss graphs for the K-Fold Cross-Validation method seem to indicate that something is happening with the samples generated by the Wasserstein GAN, since it can be seen that the curves tend to be more dispersed, compared with the curves of the other two methods and the baseline, even though the folds where class balanced in both tasks. Also, it was observed that the neural networks were more prone to overfitting when using this dataset. The reason of this may be due to low intra-class variation and notorious sound artifacts present in the synthesized files. Variations of some of the hyperparameters of the GAN and the training routine were tested to analyse if the quality of the generated sounds increased, such as decreasing the number of updates of the discriminator per one of the generator, increasing the batch size, varying the gradient penalty and learning rate. None of these changes had a really appreciable change from the subjective point of view. Tests were even carried out applying band-pass and band-reject filters to the generated sounds to see if these sound artifacts could be attenuated, however, there was no appreciable positive effect.

Another way in which this problem was tried to be turned around was by joining five one-second files, with the idea that at least one of the five files would have an acceptable quality, suitable for classifiers to recognize the desired class. This would also take advantage of another problem, which was training the GAN to generate five-second files, which turned out to be extremely expensive, computationally. Joining files had a slight positive effect when training the neural networks, which was observed by a slight increment in the evaluation metrics.

From the human perspective, the quality of the sounds generated for most of the classes is at least acceptable; some classes can be easily recognized even without knowing in advance which class they belong to. Others, however, are more difficult to recognize. Generated transient (sneezing, dog barking and clock) and high harmonic content sounds (crying babies and rooster) are the most faithful to the essence of its representative classes, from a human perspective. The generated quasi-structured sounds (rain, sea waves, fire, helicopter and the electric chainsaw), however, are more difficult to identify or are even unrecognisable. An objective quantitative measure that assessed the quality of the generated sounds, such as the inception score, could've been useful to accurately discriminate high-quality generated files from the low-quality ones. This metric could have also been useful after training the generative adversarial network and generating some samples, in order to find the best values for the hyperparameters or to set an early stoppage and save execution time. However, due to time constraints, it was impossible to explore the implementation of this algorithm.

4.3 Significance/Impact

The results obtained in this research confirm that neural networks are algorithms capable of learning patterns from both monophonic and polyphonic sound datasets, being the latter of most interest since polyphonic sounds are what normally would be encountered in common environments. Also, from what was achieved in this work it can be stated that deep generative models such as generative adversarial networks have potential to be applied to tasks involving sound. Moreover, by selecting specific classes of sounds, diverse applications could be derived for solving specific problems in specific environments (for example [30]), including systems that would do recognition of environmental sounds in real-time.

4.4 Future Work

There is much that can still be done from what has been achieved in this work. On their own, each of the specific objectives could be worked individually and improved. However, there are some points to highlight. First of all, the methodology of this work was applied in a dataset that could definitely be considered small. There are larger datasets such as the UrbanSound8k (8732 files), which originally was also planned to be tested and the results included in this work, however, due to time constraints this was not possible, since, in addition of having to repeat each step of the methodology, when carrying out some tests it was appreciated that the computational cost increased considerably due to the greater number of samples per class. This was right away evident when training the WGAN-GP with some of the classes of this dataset. Nevertheless, it would have been very valuable be able to compare the quality of the files generated after training the WGAN-GP with both datasets, specially for the repeating classes, to be able to analyse if for example there is an improvement in terms of the presence of sound artifacts that for some classes of the ESC-10 dataset are quite noticeable. In addition, the audio files included in the UrbanSound8K were captured on the outside, so they include background noise, unlike those of the ESC-10, which were recorded under controlled conditions, which makes this comparison also interesting.

Another matter that can be worked further is the extraction of other types of features to be used as input for the classifiers, since both log-mel spectrograms and MFCCs rely on the spectral content of the signal, neglecting other important characteristics such as the rate of change of the signal over time (temporal information), well captured in delta and delta-delta features, which represent the first and second derivative (velocity and acceleration) of the

slope of the signal within a local window [1].

Further research is necessary on the subject of sound generation through deep generative models, since in this work only one variant of the existing GAN models was explored and although the obtained results were satisfactory, because they allowed to increase the robustness of the classifiers for both monophonic and polyphonic tasks, it is clear this technique can be improved, being some of the obvious reasons the debatable quality of the generated samples and the observed tendency of the neural networks to exhibit an overfitting behaviour during the training, suggesting a lower degree of intra-class variability with this particular dataset, compared to the rest of augmented datasets. As it was already mentioned, something to do in the future would be the implementation of an algorithm known as inception score which is crucial to evaluate the ability of the GAN or any other method being used to generate representative samples of the class on which the network is being trained.

Finally, more experimentation can be done using other types of machine learning classifiers. This problem is not restricted to be solved through neural networks only. Regarding the use of neural networks, more sophisticated models could be implemented, although it is relevant to mention that adding more convolutional or LSTM layers lead to a more pronounced overfitting behaviour compared to the more simple models reported in this work. The reason is because adding extra layers make it easier for the neural networks to memorize the training set; another reason why it is important to reproduce the methodology of this work with a larger dataset. An hybrid architecture of a LSTM and a CNN could also be explored in order to take advantage of the attributes of each model in one single network.

4.5 Publications

During the development of this work an article called "Wasserstein GAN con penalización de gradiente para generación de sonidos ambientales" was published on the 11th volume of mexican journal "La Mecatrónica en México". The subject of this article was the implementation of the generative adversarial network used in this work. In addition to the implementation of the GAN, the results of the classification of two classes of sounds are shown and the analysis of how the increase in the number of samples through this technique improved the performance of a neural network.

CHAPTER 5

Conclusion

In this work, two audio classification tasks, Environmental Sound Classification (ESC) and Audio Tagging, were explored using one monophonic dataset for the first task and a polyphonic one made from the same dataset. Three data augmentation techniques, including a generative adversarial network (GAN) known as Wasserstein GAN with gradient penalty (WGAN-GP), were applied to these datasets in order to compare and determine if the performance of the proposed classifiers could be enhanced by implementing these methods. The results confirm, not just that classification of both monophonic and polyphonic sounds is feasible, but also that the use of this data augmentation techniques can greatly improve the ability of a classifier to solve the assigned task.

Finally, from these results it is inferred that the cause of the low performance of the WGAN-GP with respect to the other methods of data augmentation is due to the fact that some classes are not being positively affected by the application of the method, for which it is concluded that more research is needed in this matter. Repeating the experiments with a larger dataset is essential to analyse to what extent the quality of the generated files depends on the number of samples available.

Bibliography

[1] Virtanen, T., Plumbley, M. D., & Ellis, D. (2018). Computational analysis of sound scenes and events. Springer. https://doi.org/10.1007/978-3-319-63450-0

[2] K. J. Piczak (2015). Environmental sound classification with convolutional neural networks. inProc. 25th Int. Workshop Mach. Learning SignalProcess., pp. 1–6.

[3] I. Lezhenin, N. Bogach & E. Pyshkin (2019). Urban Sound Classification using Long Short-Term Memory Neural Network. 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 57-60, doi: 10.15439/2019F185.

[4] Çakir, E., & Heittola, T. (2016). Domestic Audio Tagging with convolutional neural networks.

[5] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative Adversarial Networks.

[6] Madhu, A., & Kumaraswamy, S.K. (2019). Data Augmentation Using Generative Adversarial Network for Environmental Sound Classification. 2019 27th European Signal Processing Conference (EUSIPCO), 1-5.

[7] Tsalera, E., Papadakis, A. & Samarakou, M. (2020). Monitoring, profiling and classification of urban environmental noise using sound characteristics and the KNN algorithm. Energy Reports. 6. 223-230. 10.1016/j.egyr.2020.08.045.

[8] Münzel, T., Gori, T., Babisch, W., & Basner, M. (2014). Cardiovascular effects of environmental noise exposure. European heart journal, 35(13), 829–836. https://doi.org/10.1093/eurheartj/ehu030

[9] Chandrakala, S., & Jayalakshmi, S.L. (2019). Environmental Audio Scene and Sound Event Recognition for Autonomous Surveillance. ACM Computing Surveys (CSUR), 52, 1 - 34.

[10] I., Gulrajani., F., Ahmed., M., Arjovsky., V., Dumoulin. & A., Courville. (2017). Improved Training of Wasserstein GANs. arXiv: Learning,

[11] Silva, B. da, Happi, A. W., Braeken, A., & Touhafi, A. (2019). Evaluation of Classical Machine Learning Techniques towards Urban Sound Recognitionon Embedded Systems. Applied Sciences, 9(18), 3885. https://doi.org/10.3390/app9183885

[12] Y. Petetin, C. Laroche & A. Mayoue. (2015). Deep neural networks for audio scene recognition. European Signal Processing Conference (EUSIPCO2015), Nice, France, August 31- September 4, 2015.

[13] Sachin, Chachada & C.-C., Jay, Kuo. (2013). Environmental sound recognition: A survey. 1-9. doi: 10.1109/APSIPA.2013.6694338

[14] Mushtaq, Z. & Su, S. F. (2020). Environmental sound classification using a regularized deep convolutional neural network with data augmentation. Appl. Acoust. 167, 107389.

[15] Vu, T.H., & Wang, J. (2016). Acoustic scene and event recognition using recurrent neural networks.

[16] Zhang, Z., Xu, S., Qiao, T., Zhang, S., & Cao, S. (2019). Attention based Convolutional Recurrent Neural Network for Environmental Sound Classification. ArXiv, abs/1907.02230.

[17] Esmaeilpour, M., Cardinal, P., & Lameiras Koerich, A. (2019). Unsupervised Feature Learning for Environmental Sound Classification Using Cycle Consistent Generative Adversarial Network. ArXiv, abs/1904.04221.

[18] Donahue, C., McAuley, J., & Puckette, M. (2018). Adversarial Audio Synthesis. International Conference on Learning Representations.

[19] Madhu, A. & K. Suresh. (2022). EnvGAN: a GAN-based augmentation to improve environmental sound classification. Artif. Intell. Rev. 55, 8 (Dec 2022), 6301–6320. https://doi.org/10.1007/s10462-022-10153-0

[20] Çakir, E., Heittola, T., Huttunen, H., & Virtanen, T. (2015). Polyphonic sound event detection using multi label deep neural networks. 2015 International Joint Conference on Neural Networks (IJCNN), 1-7.

[21] Puente, S. (2018). Single and Multi-Label Environmental Sound Classification Using Convolutional Neural Networks.

[22] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[23] Stein, J. Y. (2000). Digital Signal Processing: A Computer Science Perspective. 2nd ed.; John Wiley & Sons, Inc: New York, NY, USA.

[24] Piczak K. J. (2015). ESC: Dataset for Environmental Sound Classification. Proceedings of the 23rd Annual ACM Conference on Multimedia, Brisbane, Australia.

[25] al Aied D. (2019). An implementation of Wasserstein GAN to generate 5 different Gaussian distributions. Retrieved from https://github.com/dhyaaalayed/wgan-gaussian#license.

[26] Salamon, J., Jacoby, C. & Bello, J. (2014). A Dataset and Taxonomy for Urban Sound Research. Proceedings - 22nd ACM International Conference on Multimedia.

[27] J. K. Das, A. Ghosh, A. K. Pal, S. Dutta and A. Chakrabarty, (2020). Urban Sound Classification Using Convolutional Neural Network and Long Short Term Memory Based on Multiple Features. Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), Fez, Morocco, 2020, pp. 1-9.

[28] Wei, S., Xu, K., Wang, D., Liao, F., Wang, H., & Kong, Q. (2018). Sample Mixed-Based Data Augmentation for Domestic Audio Tagging. ArXiv, abs/1808.03883.

[29] E. Cakir, T. Heittola, H. Huttunen and T. Virtanen. (2015). Multi-label vs. combined single-label sound event detection with deep neural networks. 23rd European Signal Processing Conference (EUSIPCO), Nice, France, 2015, pp. 2551-2555.

[30] Zhang, L., Towsey, M., Xie, J., Zhang, J., & Roe, P. (2016). Using multi-label classification for acoustic pattern detection and assisting bird species surveys. Applied Acoustics, 110, pp. 91-98.

[31] Baelde, M., Biernacki, C., and Greff, R. (2019). Real-Time monophonic and polyphonic audio classification from power spectra. Pattern Recognition, vol. 92, pp. 82–92.