Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Study, Implementation and Evaluation of Event Detection and Anomaly Identification Systems based on acoustic information

Author: Daniel Bermejo Llorente

Advisor: Javier Macías Guarasa

Co-advisor: Juan Manuel Vera Díaz

2021

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Study, Implementation and Evaluation of Event Detection and Anomaly Identification Systems based on acoustic information

Author: Daniel Bermejo Llorente

Advisor: Javier Macías Guarasa

Co-advisor: Juan Manuel Vera Díaz

Tribunal:

President: Juan Carlos García García

1st Vocal: José Manuel Villadangos Carrizo

2nd Vocal: Javier Macías Guarasa

Deposit date: October 7th, 2021

A mi familia, amigos y amigas, porque saben lo dura que ha sido esta etapa y han estado ahí para apoyarme. Este cierre es tan mío como vuestro.

"Empieza haciendo lo necesario, luego haz lo posible y de pronto empezarás a hacer lo imposible."
Francisco de Asís

Acknowledgements

"Un viaje de mil millas empieza con un solo paso".

Lao-Tse

Este proyecto ha sido el resultado de muchas horas de esfuerzo y dedicación. Ha tenido sus altibajos, con más bajos que altos, pero la satisfacción de poder llegar a la cima siempre tapa todo lo malo que ha venido delante. Sin embargo, ésto solo es así si hay un grupo de personas detrás que está ofreciendo su comprensión, apoyo y amor incodincional, estando siempre a tu lado, pase lo que pase.

Por ello, le quiero agradecer a mi familia su fé ciega en mí y en mi capacidad para sacar adelante lo que ha sido una de las etapas más duras de mi vida, aunque también la más enriquecedora y la que más alegrías me ha dado. Mis padres y mi hermano han sido los pilares que me han llevado a estar donde estoy ahora, porque sin ellos no habría sido capaz de seguir en muchas ocasiones, haciéndome ver que podía dar otro paso cuando yo no me veía capaz. Ellos son mi gran referente de esfuerzo y dedicación como medio para obtener todo lo que uno se propone.

También se lo quiero agradecer a todos los amigos y amigas que me han acompañado durante estos años. Habéis sido un gran muro detrás de mí, que me ha ayudado a seguir avanzando con seguridad. Son las personas con las que más he compartido mis preocupaciones y sufrimiento, buscando un apoyo que me ayudase a levantarme tras cada caída y encontrando un trampolín, que no solo me levantaba sino que me catapultaba hacia el siguiente objetivo.

Por último, le quería agradecer a los profesores que tanto me han enseñado y aportado a lo largo de los años que han durado mis estudios de grado. Yo entré en esta carrea por una recomendación y acabé enamorado de las telecomunicaciones. Esto es, en parte, gracias a esos profesores que saben transmitir no solo su conocimiento en la materia, sino también su admiración por ella. Una especial mención a mi tutor de TFG, Javier Macías Guarasa, a parte de por ser un tutor comprensivo y siempre dispuesto a ayudar, por el profesor que fue, porque hizo de una asignatura complicada un entretimiento.

Por todo esto, gracias.

Resumen

En la actualidad, el interés por la detección de eventos anómalos ha ido en aumento entre diferentes campos de investigación del estado del arte, como la visión por ordenador, el procesamiento de señales, la banca, etc. Las técnicas de Machine Learning (ML), y en concreto las técnicas de aprendizaje profundo, o Deep Learning (DL), han tenido un gran impacto en el desarrollo de las recientes aproximaciones, permitiendo grandes mejoras en cuanto a los índices de precisión de los sistemas propuestos. La visión por ordenador es el campo más avanzado en esta área. No obstante, existen sistemas en los que este problema se aborda a través de la información acústica proporcionada por un micrófono, o un cojunto de ellos, colocado en un entorno, debido a diferentes condicionantes: *i)* Privacidad del usuario; entornos en los que se debe monitorizar una situación y avisar si se encuentra alguna anomalía. Un ejemplo de este tipo de sistema es un sistema de detección de violencia doméstica desplegado en un hogar. *ii)* Mal funcionamiento de maquinaria; Componentes como el interior de un motor en donde es complejo instalar una cámara para comprobar el desgaste de las piezas o su correcto funcionamiento, abordar esta tarea con información acústica es una solución típica

A partir de un estudio del estado actual del arte en la detección de eventos acústicos anómalos, se ha considerado utilizar un sistema existente para el desarrollo de este trabajo fin de grado. Los principales objetivos planteados han sido: reproducir los experimentos realizados por los desarrolladores del sistema elegido, consiguiendo así resultados similares; cambiar la base de datos utilizada para entrenar, validar y probar el sistema, con el fin de estudiar la adaptabilidad de la red a un nuevo tipo de datos; y modificar la red dada para estudiar el efecto que éstas tienen en el rendimiento del sistema.

Además, se ha estudiado un segundo sistema. Dicho sistema, denominado SELDNet, es bien conocido en el estado del arte y se centra en la detección de eventos acústicos así como en la clasificación multiclase de los mismos. Aunque no se aproxima a la tarea de detección de eventos anómalos propuesta en este proyecto, es relevante su estudio ya que un primer paso para la detección de anomalías es la detección de los eventos acústicos.

Palabras clave: Detección de eventos sonoros, Anomalías, Aprendizaje profundo, Keras, Tensorflow.

Abstract

Nowadays, the interest in detecting anomalous events has been rising within different state-of-the-art research fields, such as computer vision, signal processing, banking and so on. Machine Learning techniques, and specifically Deep Learning techniques, have had a great impact on the recent approaches developed, allowing great improvements in terms of the accuracy rates of the proposed systems. Computer vision is the most advanced field in this area. Nevertheless, there are systems where this problem is addressed through the acoustic information provided by a microphone placed inside an environment, due to different constraints: *i)* User privacy; environments where a situation must be monitored and a warning given if an anomaly is found. An example of this kind of system is a domestic violence detection system deployed in a house. *ii)* Machinery malfunction; Components such as engines where it is complex to set up a camera inside to check the wear of the pieces or their correct operation, approaching this task with acoustic information is a typical solution.

Based on a study of the current state of the art in the detection of anomalous acoustic events, it has been considered to use an existing system for the development of this degree final project. The main objectives set have been: to reproduce the experiments carried out by the chosen system developers, thus achieving similar results; to change the database used to train, validate and test the system, in order to study the adaptability of the network to a new type of data; and to modify the given network to study the effect that these have on the performance of the system.

In addition, a second system has been studied. Said system, named as SELDNet, is well-known in the state of the art and focuses on the detection of acoustic events as well as on the multi-class classification of them. Although it does not approach the anomalous event detection task proposed in this project, it is relevant to study it since a first step for anomaly detection is the detection of the acoustic events.

Keywords: Sound event detection, Anomaly, Deep learning, Keras, Tensorflow.

Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial Neural Network.
AUC	Area Under the Curve.
CNN	Convolutional Neural Network.
CRNN	Convolutional Recurrent Neural Network.
DCASE	Detection and Classification of Acoustic Scenes and Events.
DL	Deep Learning.
FN	False Negative.
FP	False Positive.
Leaky ReLU	Leaky Rectified Linear Unit.
ML	Machine Learning.
MSE	Mean Squared Error.
NN	Neural Network.
pAUC	partial Area Under the Curve.
PReLU	Parametric Rectified Linear Unit.
ReLU	Rectified Linear Unit.
ROC	Receiver Operating Characteristic.
SED	Sound Event Detection.
TN	True Negative.
TP	True Positive.

Chapter 1

Introduction

1.1 Motivation

In the analysis of human environments, event detection is a fundamental goal in different areas of service automation (such as security, health-care, marketing and sales, and so on). It is usually based on the multimodal data available through the large deployment of intercommunicating sensors and the increasingly powerful processing systems.

Many of the state-of-the-art approaches make use of footage from surveillance networks to detect different events [1]. However, there is an increasing interest in developing systems capable of the same things by making use only of acoustic information, due to limitations imposed by some systems on installing cameras on it.

The identification of acoustic phenomena has caught the curiosity of the scientific community [2–4], especially those that can be considered as anomalies. Anomalous events are commonly defined as those that occur infrequently in comparison to a normality model [5]; or those that have distinct characteristics from those that are described as normal; or those that have a specific significance based on an abnormality model of interest [6].

The project focuses on the improvement, usage and assessment of an anomaly detention framework based on acoustic information. It seeks after the integration of high level acoustic data in mixed media occasion location frameworks pointed at the location of irregularities or unsafe circumstances, such as physical or psychological hostility circumstances [7, 8], where audio is usually the main or only source of data.

In this final degree thesis, a system developed for this purpose is studied to test its capabilities when addressing such a task.

Additionally, an event classification system will be initially analyzed, as an study of how well actual systems are performing in the task of identifying different acoustic events.

1.2 Objectives

The main objective of this project is the design, implementation and evaluation of systems capable of detecting acoustic events, oriented to the identification of anomalous situations, only using acoustic information. The project analyzes available systems and databases and generate a complete implementation which is evaluated and compared with the results of proposals available in the literature.

The work involves the use of audio signal processing techniques, combined with those of machine learning, and the application of rigorous strategies of experimentation and algorithmic validation. The development environment is based on a GNU/Linux platform, mainly using the Python programming language.

As the general methodology we are applying, we start by selecting an existing deep learning based system, capable of detecting if an audio signal has anomalous events, and then performing different experiments and modifications, studying how well it performs. The major stages for this work are:

- Get the initial existing system to work
- Reproduce the results achieved by the initial existing system team by conducting the same experiment with the same data-set they used. This system is based on an encoder-decoder architecture.
- Evaluate the system performance when addressing a different data-set that will be built ad-hoc to better fit our final purpose (acoustic anomaly detection in human environments).
- Modify the structure and working parameters of the system in order to determine their effect in further improving the results.

Finally, we will also address the initial evaluation of an acoustic event classification system, to set the basis for future work in the anomaly detection task, but using an approach different from the encoder-decoder architecture discussed above.

To accomplish all these objectives, it has been necessary to address several intermediate tasks:

1. Get training in machine learning, focusing on deep learning technology and its related tools.
2. Make a detailed study and inventory of the availability of tools, systems and databases so the most appropriate for our main objective could be used.
3. Choose, from the compiled inventory, one network focused in anomaly event detection and a second system for the event classification task.
4. Take the anomaly detection system, make it run in the laboratory computer and obtain its results when using the default configuration and data, discussing the results and comparing them with those obtained by the developer team of the network.
5. Generate a new database, mixing different data-sets, and use it to test the anomaly detection system, analyzing its adaptability to a different environment.
6. Modify the structure and working parameters of the anomaly detection system and study if the results obtained with the data-set created can be improved.
7. Run the event classification network and perform a study of the results, comparing them with those obtained by the developer team.

1.3 Document structure

The final thesis is divided into 4 different chapters detailed next:

Chapter 1: Introduction: The current chapter, that sets out the main objectives of the work, as well as the organization of the following chapters.

Chapter 2: Prior work. In this chapter, an introduction to the Sound Event Detection (SED) field and to the main concepts of Artificial Neural Network (ANN) is made, to set the adequate context for the work.

Chapter 3: Implementation and results.

In which the networks under study are defined, as well as the different experiments performed with each one, including a final discussion on the results obtained with them.

Chapter 4: Conclusions and future work.

In which we outline the main conclusions of our work, also introducing some possible future lines for continuing this study.

Chapter 2

Prior work

2.1 Introduction

In this chapter, concepts related to acoustic event detection or neural networks are provided (most of them extracted from [9], which was a course taken by the author to prepare for the final thesis work. Thus, the main ideas of Sound Event Detection (SED) and the definition of anomalous events are given, as well as a general overview of the structures under study and the concepts needed to understand them. Also, we make a quick review of some state-of-the-art systems and databases raised as suitable for the project and the metrics used.

2.2 Sound Event Detection

Our environments consists of very complex mixtures of audio signals, since there are always several sound sources at the same time. SED is the task of identifying sound events in a recording and their respective temporal start and end times [10]. It comes naturally to humans to recognize different events, even without the help of visual support, just processing the acoustic information received. We have the capacity to sort out the different audio signals and focus on the source of interest. This is also related to how we are able to *solve* the *cocktail party effect*, referring to the capacity we have in parties to focus on a conversation despite the ambient noise and simultaneous conversations being carried out [11].

However, when it comes to computers, this task is not trivial. Despite the existence of various tools for SED, Machine Learning (ML), and specially Deep Learning (DL), has given a big push to the topic. Even so, it has taken years of research on "training" the encountering and learning correlations between the massive diversity of sounds that are found in everyday situations. Currently, this is mainly implemented thanks to deep learning techniques that require a great computational power.

Nowadays, detecting different events in an acoustic scene can improve the time of response of the authorities, as a surveillance system [12, 13], or even prevent catastrophes from happening, since the identification of events such as screams or abnormal sounds in an engine could send alerts even before anything actually happened. Due to the great sophistication this task requires, most of the current systems are implemented using DL techniques, that are one of the most complex approaches to ML.

2.2.1 Anomaly event detection

This project seeks as its main objective to develop a network capable of detecting anomalous events from acoustic information. The concept of anomaly is complex to defined, since it depends on the situation and context.

For this thesis, an anomaly is defined as the acoustic data with characteristics that differ from the normality model of the information provided in the databases used. To be more precise, it will be considered that an anomalous event has occurred when atypical or unusual sounds are detected. For example, in a recording of the street, a car passing by is a *normal* sound event, while its breaks squeaking will be considered *anomalous*. Another example could be the sound of a toy car machinery when it is working properly (*normal* event) versus the sounds made when some part of it is broken (*anomaly*).

2.3 Artificial Neural Networks

Artificial Neural Network (ANN), also known as Neural Network (NN), used in many ML applications, are systems modeled after the biological neural networks that make up animal brains. They are computational learning systems that understand and convert data input into a desired output using a network of functions, called artificial neurons, which try to reproduce the neurons behaviour in a biological brain.

Neurons are computing units that multiply the input they get by coefficients, also known as *weights* and *biases*, giving a greater or smaller importance to one input or another. These weights and biases are recalculated in every training iteration, so the model generated by the network more closely resemble the input data. Their initial values can be set by he developer to provided values or randomly. The calculation of the neuron considering weights and biases is as follows:

$$y = \sum_{i=0}^{N-1} \omega_i x + b_i, \tag{2.1}$$

where y is the neuron activation, x is the input source, ω_i is the i^th weight with $i = 0, 1, \ldots, N-1$ and b_i is the i^th bias.

After that, the neurons generate a final output value after applying a so called *activation function*.

This function let the system apply a non-linearity to the neuron output so that the system can be complemented in a more complex way, since most of the problems nowadays are not linear. In a simple way, activation functions are responsible for the neuron activation, and decide to what extent the neuron is activated. An example of an Artificial Neural Network (ANN) is shown in Figure 2.1.

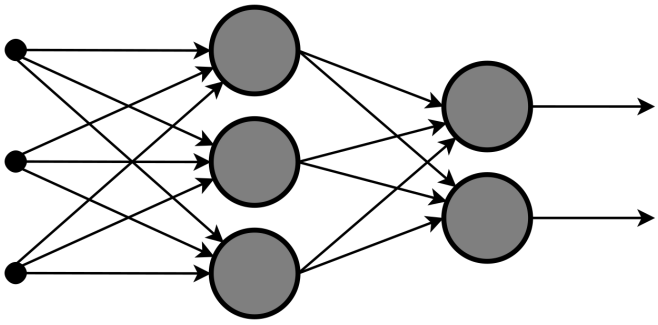


Figure 2.1: Simple Artificial Neural Network [14].

The connections between neurons, known as *edges*, allow for signal transfer between neurons, the same way the *synapses* works in biological brains. When signals are received by a neuron, they are processed and the neuron can send the resulting signals to those connected to it. The output signal of a neuron is obtained by the activation funcion described above.

In the specific case of DL, neurons are organized into multiple layers, so they are only connected to others in the previous or following layer. Thus, the layer receiving the input external data is known as the *input layer*, the one producing the final result is the *output layer*, and the ones in between them are the *hidden layers*. There can be none or more of these hidden layers as shown in Figure 2.2.

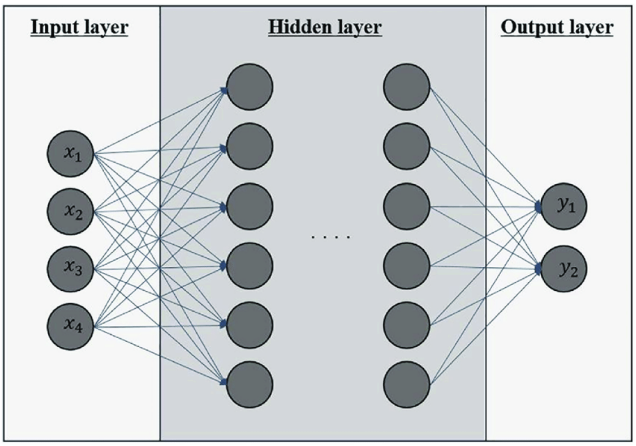


Figure 2.2: Layers in a Neural Network [15].

ANNs are increasingly being used for machine learning algorithms since there is no need of programming what are the operations to be carried out by the network, as it is able to learn by processing labeled data. Thus, two types of input data can be distinguish: *training data* and *evaluation data*. The first one is the data used for training the network. This data is basically an input that also specifies the output expected to be generated, so that the NN learns the key characteristics of the input to generate the correct output. The second one, the evaluation data, is data never seen before for the network, so that it will allow us to test how well the network performs when generating its output values. These outputs can be a classification of events, detection of anomalies, etc.

From these two different data types, two phases can also be differentiated: the *training phase* and the *evaluation/test phase*. During the training phase, the network extracts the important characteristics of the training data, it *learns* how the data is, and generates outputs, improving the results in every iteration and generating a model in which it is specified the values of the weights in each neuron to generate the best results. Successively, part of the training data is used for validating the model the network is making after the input data. It is used to check if the network is still improving or it is just fitting too much to the training model and will perform poorly with never-seen-before data.

The algorithm employed for training NNs is known as Stochastic Gradient Descent (SGD). It compares the output given by the network with the labeled output expected, computing an error between them. This error is propagated from the network output layer to its input layer, modifying the weights in each layer proportionally to the amount of its contribution to the final. This is called *backpropagation* and it is executed in every iteration of the training phase. Each iteration takes a subset of data, known as *batch*, from the whole data-set used for training. The size of the batches is defined as a parameter for the system. When the whole data-set is processed by the network, it is called an *epoch*. Once an epoch is finished, this algorithm is executed as many epochs as the developer specifies to the network.

Once the training phase is finished, the NN is supposed to be able to receive new data, and successfully generate accurate results. That is where the evaluation phase starts, when the new data (*evaluation data*) is given to the network and an error rate can be obtained, based on the output generated.

The bigger and more varied the training data is, the more accurate the output of the ANN will be. However, it can be counterproductive to introduce a big amount of training data, since the network can fit to much to an specific model of training data and it will not be able to robustly generalize to unseen data. This is known as *overfitting* [16]. In order to avoid this problem, the training data must also be varied, and the network size should be big enough. In Fig. 2.3, the same model is shown when using too much training data (*Overfittied*), the right amount (*Good Fit/Robust*) or not enough (*Underfitted*). The dashed line indicates the guesses made by the network when identifying the events, described as *values* in the figure and represented as dots. When too much data samples are given to the network, it precisely guess them, although with unseen data will fail drastically since it learned the characteristics of the training data-set, not generalizing the model. On the contrary, with not enough samples, the system performs poorly when guessing. It is difficult to establish the right number of samples to obtain a well-generalize and robust model.

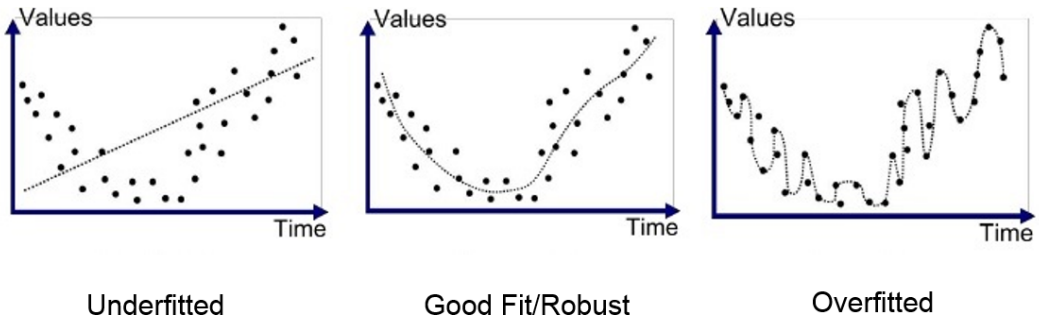


Figure 2.3: Overfitting illustration [17].

Some neural network architectures are explained below, as well as different techniques and operations used to improve their performance.

2.3.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) are a type of ANN designed to work in a manner similar to that of neurons in the primary visual cortex of a human brain, since we are very efficient classifiers, and most neural network architectures try to imitate it. They have been found to be very effective in image classification and segmentation tasks. They consist of a multitude of convolutional filter layers of one or more dimensions, behind which nonlinear activation functions are inserted.

In its classification process, 2 phases can be distinguished:

Feature extraction: This is the initial phase and is mainly composed of convolutional neuron layers that resemble the processing of the human visual cortex. The more you advance through the number of convolutional layers, the less they react to the variation of the input data and the greater is the abstraction achieved by them to recognize more complex shapes. So, for example, when extracting the features of an image of a car, in the first layers the network will detect the edges or the color of the car, while in the last layers there may be a filter that defines the complete shape of a car.

Classification: They are based on the use of dense layers, which are a collection of fully-connected neurons between each of them. This phase allows us to relate all the extracted characteristics and to carry out the classification.

This kind of networks allows to relate the complete set of characteristics of a model and classify them according to that relation.

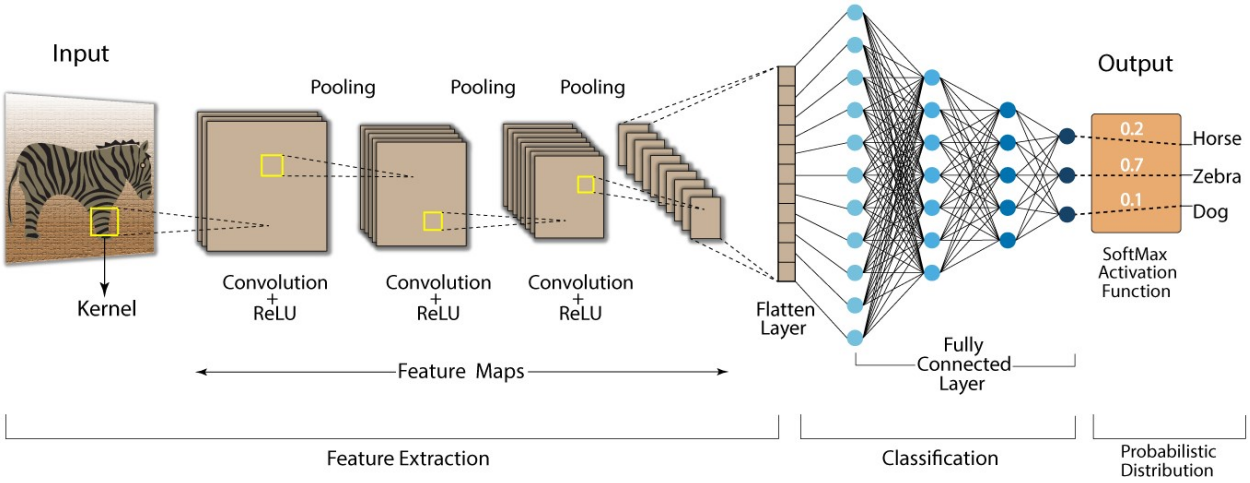


Figure 2.4: CNN schema [18].

In this project, a Convolutional Recurrent Neural Network (CRNN) is used. The main difference between CNNs and CRNNs is that the second ones include recurrent layers in their architectures. This let them relate previous information to an event to the event itself, giving it *memory*, the same way a human relates a previous conversations to a discussion to the discussion itself. In particular, the network tested had Gated Recurrent Unit (GRU) layers (Fig. 2.5).

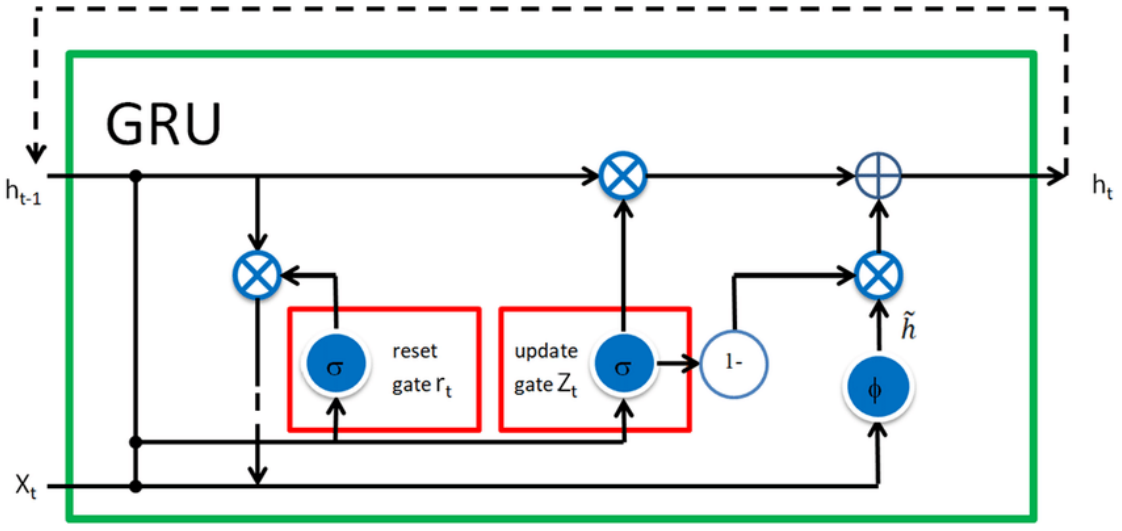


Figure 2.5: GRU schema [19].

This kind of neurons have 2 types of inputs. The first one is those inputs that take the output from the previous neurons (X_t), and the other (h_{t-1}) takes the output generated by itself in previous iterations (previous information). Internally, it is composed of two gates, the *reset gate* (r_t) and the *update gate* (Z_t), in charge of deciding how much of previous information can be forgotten and how much is important to keep so it influences to the new output, respectively.

2.3.2 Autoencoders

Autoencoders are a specific type of ANN, composed of a non-linear encoder and a decoder. This kind of networks has proven to be suitable for the anomaly detection kind of tasks proposed in this thesis [20,21].

Autoencoders are trained to *copy* the input given to its output by first encoding the input (compressing it) using a non-linear function (e.g. f) and then decoding it using a second non-linear function (e.g. g). The training criteria is to obtain the original data as accurately as possible, as seen in Figure 2.6. Its objective is to minimize the differences between input data and the output, so it improves how the compression of the inputs is perfomed and the reconstruction error is reduced [22].

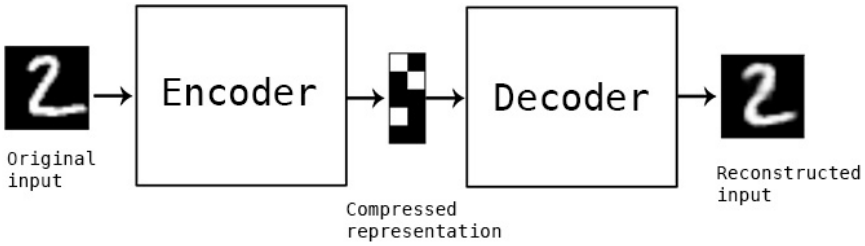


Figure 2.6: Autoencoder schema [23].

Auoencoders ar trained with normal data so they generate normality models of the data. The way autoencoders differentiate whether the data is normal or anomalous is by comparing how well the input is reconstructed at the output. Since the network is only trained with normal data, when an anomalous sample is introduced, it will fail at the reconstruction, or at least it will be worse than those generated when a normal audio is given to the network. Thus, it results in a bigger reconstruction error when the file inserted does not contain an anomaly. Comparing the reconstruction error with a maximum threshold error, the system makes the binary classification of anomaly or no-anomaly

2.3.3 Activation Functions

As explained before, activation functions establish how and when every neuron is activated. In a more technical way, activation functions define how the sum of the weighted inputs is converted into an output in every neuron of every layer. Thus, different activation functions can be used, one per layer the system has. Normally all hidden layers use the same activation function, and the output layer a different one, depending on the network task [24].

Choosing one or another may have a great impact in the capability and performance of the network, as it will influence in the predictions made by a trained model. In this project, the main activation function used is the Rectified Linear Unit (ReLU) function. We also evaluated the use of additional ones: Parametric Rectified Linear Unit (PReLU) and Leaky Rectified Linear Unit (Leaky ReLU), which are explained below.

2.3.3.1 Rectified Linear Unit Activation

ReLU function [25] is the most used activation function nowadays in deep learning since it makes easier to train systems and normally achieve good enough performance. It directly outputs the weighted inputs value when it is positive, or zero when it is negative (Figure 2.7), and it is defined as:

$$f(y) = \begin{cases} y & \text{if } y > 0 \\ 0 & \text{if } y < 0 \end{cases} \tag{2.2}$$

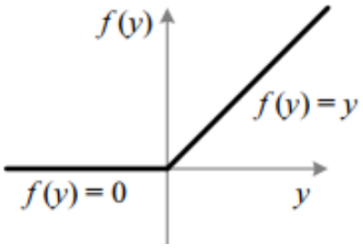


Figure 2.7: ReLU activation function [26].

By doing this, since there is no need for updating the weights when the result of the weighed input addition is negative, it leads to the disconnection of some neurons and, thus, it may reduce overfitting by stopping the evolution of the learning process.

2.3.3.2 Leaky Rectified Linear Unit Activation

Leaky ReLU is a modification of ReLU which, instead of giving an output of zero when the weighted input was not positive, it is also linear, with a low slope (Figure 2.8), referred to as *leak*. This slope is usually as small as $0.01 - 0.02$, and it has the following expression:

$$f(y) = \begin{cases} y & \text{if } y > 0 \\ \alpha y & \text{if } y < 0 \quad \text{with} \quad 0.01 < \alpha < 0.02 \end{cases} \tag{2.3}$$

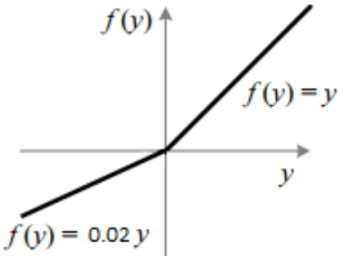


Figure 2.8: Leaky ReLU activation function [26].

With this, instead of reaching a dead end, neurons are not completely deactivated and keep evolving. However, according to some studies, Leaky ReLU can provide worse results and performance [27].

2.3.3.3 Parametric Rectified Linear Unit Activation

PReLU is similar to the previous one. In this case, it has a greater impact in the performance, since the slope for the weighted input negative values is bigger, and so the learning speed of neurons is increased with nearly zero extra computational cost [26]. This activation function is defined as follows:

$$f(y) = \begin{cases} y & \text{if } y > 0 \\ \alpha y & \text{if } y < 0 \quad \text{with} \quad \alpha \in \mathbb{R} \end{cases} \tag{2.4}$$

This kind of activation function is more likely to fall into overfitting.

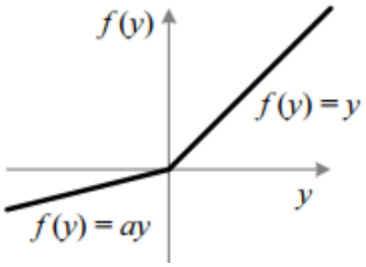


Figure 2.9: PReLU activation function [26].

2.3.4 Regularization techniques

When the modeled trained by a NN becomes more and more complex, it can reduce the training error drastically. However, the validation error not always follows that reduction and can even start to increase. It means that the network is fitting too much to the training data and will result in poor performance with unseen data. In other words, it falls into overfitting.

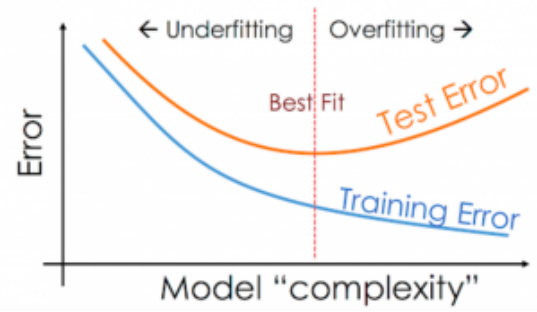


Figure 2.10: Error evolution [28].

Regularization techniques are used in ANN to prevent this from happening, and to stabilize the learning process. This techniques *freeze* and reduce the evolution speed of the weights in the neurons, meaning some of them are stopped from learning and progressing their coefficients or doing it slowlier.

There are many regularization techniques, and we are going to explain three of them, as they have been used in the thesis implementation.

2.3.4.1 Early stopping

This is one of the most common techniques used in network training. When training a large model on a big enough data-set over an extended period of time, rather than boosting the model's generalization power, it will increase the likelihood of overfitting. The training error will continue to be reduced but, at certain point, the validation error will start increasing rather than falling. At that point, it will have started to overfit.

The main idea behind early stopping consists of halting the updating of the network coefficients when the validation error starts to increase. It can also store the values of the weights when a minimum in the training and validation error is reached, so it is considered as the best model up to this point. Usually, and it is that way in the systems developed in this project, the network performs a few more iterations, specified as a parameter by the developer, to check if a better model can be trained. If not, the training is completely stopped and finished, independently of whether the number of epochs has been reached or not, considering the last saved model as the best one and using it for the evaluation process.

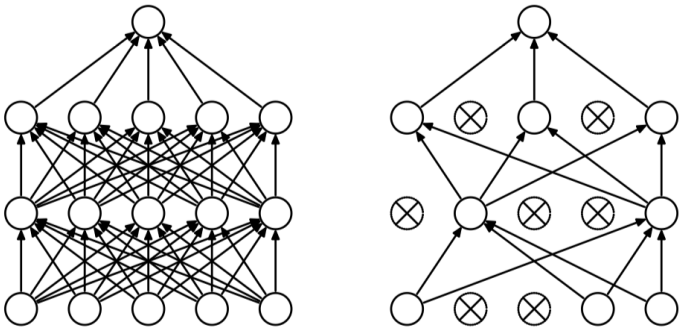
2.3.4.2 Batch normalization

This technique comes from the need for coordination between the high amount of layers in deep networks, due to the randomness of the initial values of the weights and since every layer's neurons take as input the previous layer's outputs. In these networks, a small change in the initial layers may involve a huge amplification of the change when it propagates to deeper layers.

It consists of standardizing the input in every layer so the learning process is stabilized, meaning the coefficients in the neurons do not make huge changes in their values, so the number of iterations needed to train deep networks is reduced. Besides, this balancing of network learning improves the generalization capabilities of the system, so that the trained model created does not fit too much to the input data, thus reducing the risk of overfitting [29]. This normalization is performed per batch, calculating its mean and deviation to make the standardization, not to the complete data-set used.

2.3.4.3 Dropout

As anticipated in subsection 2.3.3.1, a good way to reduce the effect of the overfitting is *switching off* neurons, so that they do not progress in their learning (Figure 2.11). That is the idea behind this technique: unique combinations of the neurons are being ignored or set to zero randomly at different points in time during the training phase. It is not exactly the same process done by the ReLU activation function. When applying dropout, independently of the outcome of the operations performed by the neurons, the units to which it is applied, are stopped from evolving their coefficients and they do not generate any output. In several studies, it has been observed that, at any given moment in time, the models with dropout had a lower classification error than the same models without dropout [30–32].



(a) Standard Neural Net (b) After applying dropout.

Figure 2.11: Example of dropout application in a system [33].

Thus, dropout helps minimizing the overfitting on the training data by lowering the reliance of each neuron on other units in layers, resulting in a smaller error. This suggests that dropout regularization helps to break the co-adaptations among neurons, allowing each neuron to function in a more autonomous way.

2.4 Revision of state-of-the-art systems

With all this concepts assimilated, a extensive revision of state-of-the-art systems has been made, searching for anomaly detection oriented networks. Currently, most of the systems are based on visual information or a mix of visual and acoustic data. However, since it is gaining interest and importance to find solutions to event detection in environments where is not possible to place cameras, it is not difficult

to find systems only based on acoustic information. Moreover, there are a group of challenges called Detection and Classification of Acoustic Scenes and Events (DCASE), in which many different research groups develop their own systems to accomplish given tasks proposed in each year's challenges. Those tasks consist of the detection and/or classification of events and scenes, sometimes anomalies (depending on each years challenge's tasks) only using the audio data-sets the organizers provide. After making a wide research looking for networks to use, taking into considerations the complexity of the systems and in order to make it easier, the final revision was made focusing in the DCASE challenges from 2017 until now.

2.4.1 Libraries

When analysing suitable systems for this thesis, it is important to choose those using Python [34] as coding language and the libraries Keras [35] and TensorFlow [36] as construction and training libraries for the models. It is decided like that since most of the systems developed in our research team were based on them, so it would be easier to give support when incompatibility errors would appear, apart from being the most used libraries in DL nowadays.

The reason to use a network coded in Python is that it is the most widespread programming language in this area. It allows developers to write concise and readable code, thanks to its simplicity, while it is executing complex algorithms for machine learning. Moreover, it is easy to learn since is very understandable by humans. This makes it more intuitive than other programming languages. In addition, there are many libraries and frameworks developed for machine learning in python.

2.4.1.1 TensorFlow

TensorFlow is a library developed by Google Brain for its machine learning applications and deep neural networks. It is a mathematical computation library, which quickly and efficiently executes flow graphs. A flow graph consists of mathematical operations represented over knots, and whose input and output is a multidimensional vector (or tensor) of data.

Right now, there are two different TensorFlow versions: TensorFlow 1 and TensorFlow 2. The main difference between them is that TensorFlow 2 makes transparent for the programmer the usage of the sessions in the GPU and adds an eager mode to the tensors, making possible to debug in real time. Moreover, Keras is embedded in TensorFlow 2, giving it a greater importance and making it more efficient to train the models. For that second version, Keras is imported from TensorFlow, instead of importing it independently. This did not have any impact in the development of the project, it was indifferent during the system search, as both versions are easy to install and use. In the end, the version used to run the networks under study was 1.15.2.

2.4.1.2 Keras

Keras is a neural network library written in Python, also developed by Google. It is a high-level API for the creation of learning models. It provides a homogeneous syntax and a simple, modular and extensible interface for the creation of neural networks. (Version 2.1.6)

2.4.1.3 NumPy

This library is very commonly used. It is a Python mathematical library for big multidimensional arrays and matrices, giving a large collection of mathematical functions to operate with them. (Version: 1.20.2)

2.4.1.4 Pandas

It is a library made for the manipulation and analysis of data structures and time series, as an extension of NumPy library. (Version: 1.2.4)

2.4.1.5 Scikit-Learn

Library containing several classification algorithms, using NumPy for the operations support. (Version: 0.20.2)

2.4.1.6 LibRosa

Python library for the analysis of audio record's information. (Version: 0.7.2)

2.4.1.7 TQDM

This is just a graphic library for the display of progress bars in Python. (Version: 4.43.0)

2.4.2 Review of the evaluated networks

As a main objective, the first system to be chosen had to be an anomaly event detector network. For doing so, the tasks related to anomaly identification in the DCASE challenges were considered. From the existing challenges since 2017, only in two of them we could find tasks related to this topic: task 2 of DCASE 2017 and task 2 of DCASE 2020.

Finally, in an attempt to obtain the most current networks, the networks considered were taken from the DCASE 2020 challenge task 2. This task consisted of detecting anomalous sounds in machine monitoring environments. The detection of anomalies in this machines is not the objective of the project, but it is good starting point for the study performed. After all a new data-set is created in a second experiment, which is more orientated to the detection of anomalies in human environments.

Thus, three systems were pre-selected for the anomaly detection task:

1. Autoencoder baseline system from the task [37].
2. Improved baseline autoencoder system [38]. Ranking place as a team: 35. Ranking place of the system: 87 (Based on te results provided).
3. Dense autoencoder [39]. Ranking place as a team: 24. Ranking place of the system: 58 (Based on the results provided).

All of them fulfilled the criteria for choosing them for the study, so, in the end, the last of them was chosen for being the one providing the best results of the three. The rest of the systems in the task, obtaining better results and having a superior place in the ranking, were using different coding languages and/or libraries such as PyTorch, similar to Tensorflow, or just were not sharing the code, so they were discarded.

As part of the project objectives, it was stated that a second system would be studied for the task of classifying sound events. For this, as a more secondary goal, the well-known system in the state of the art, SELDNet, developed by Sharath Adavanne and his team, a CRNN network. More precisely, the selected one was the system version they presented for the Task 3 of DCASE 2020 [40].

Apart from these networks, additional ones were analyzed, but in the field of acoustic scene classification, which consists of classifying the set of events as one. As an example, several events such as cars, large vehicles, people walking and children laughing can be classified as a street scene; as second example, events of washing machine, microwave and water boiling could be considered a kitchen scene. It was not part of the objectives, but, in case of not finding an event detector suitable for the project specifications, the next step in event classification would be this one, classifying the whole scene. From those systems, the part of identifying the events could have been isolated and used for the project purposes.

2.4.3 Databases search

Once an anomaly detection network was chosen and proven to work, there was a second task of proving the capabilities of the network with different datasets. For that purpose, an study was conducted to find suitable databases, capable of demonstrating how well the network could adapt to them.

For doing so, the possible datasets needed to have events that could be considered normal or anomalous or, alternatively, find separate datasets of normal events and datasets of anomalous events, to be later fused in an unique dataset for the evaluation process.

After carrying out this research, the following data-sets were selected as possible candidates for our objectives:

FSD50K [41] This database was created with the intent of having a large data-set for tasks related to SED, since most of the existing ones are too small. There is one really big though, the AudioSet data base, a massive amount of audio tracks taken from YouTube videos with more than 500 classes. The problematic with this dataset is that it is not an open dataset, which is a big limitation. Therefore, the creators of the FSD50K dataset took the original videos, downloaded the audio tracks and labeled them manually, generating an open dataset of over 51k audio tracks and over 200 classes.

SONYC-UST-V2 [42] This data base consists of over 18K labeled audio recordings from the *Sounds of New York City* acoustic sensor network. Those labeled events are:

- engine
- machinery impact
- non-machinery impact
- powered saw
- alert signal
- music
- human voice
- dog

TUT Sound Events 2017 [43] This data-set consists of 3-5 minute recordings from street scenes in different environments of interest for sound event detection concerned with human activity and dangerous situations. The following events are specified:

- brakes squeaking
- car
- children

- large vehicle
- people speaking
- people walking

TUT Rare Sound Events 2017 [44] This data-set comprises isolated sound events for each target class as well as background recordings of everyday acoustic situations. It also consists of 3-5 minute audio tracks. 3 different classes were defined:

- Gunshot
- Baby crying
- Glass breaking

Most of the audio tracks, when the events were not occurring, were background recording taken from the TUT Acoustic Scenes 2016 data-set. The advantage of this database is that it was already focused on anomaly detection, and so the events specified are considered anomalous.

TUT Acoustic Scenes 2017 [45] This data-set contains recordings from a variety of acoustic scenes, each with its own unique recording locations. A 3-5 minute audio recording was taken at each recording location. The original recordings were then divided into parts of 10 seconds each. This data-set was taken in case there was the need to use the scene classifier, but it was not the case.

For simplicity, recordings from the TUT 2017 database (except for the acoustic scenes data-set), were the chosen ones to be mixed and used as part of the second phase of the work in this thesis. They were both given as the data-sets of the DCASE 2017 challenge's task 3 (*Sound event detection in real life audio*) and 2 (*Detection of rare sound events*) respectively [46].

2.5 Evaluation Metrics

In this section, the evaluation metrics used to evaluate the network performance are explained.

For the anomaly event detector network, the organizers of the DCASE challenge established the area under the Receiver Operating Characteristic (ROC) curve and its partial area, as the evaluation metrics. That curve is a graphical representation of sensitivity versus specificity for a binary classifier system as the discrimination threshold is varied. This curve is constructed by plotting of the true positive rate against the false positive rate (Fig. 2.12).

In classification problems, a True Positive (TP) means the event was classified as the correct one (it was classified as X and indeed it was X), a True Negative (TN) means the event was correctly rejected to be a different event (it was determined no to be X and it was Y). On the other hand, when the received classification decision says an event was not X, but it actually was X, it is called a False Negative (FN); and when it happens to classify it as X when it was Y, it is called False Positive (FP).

Thus, the true positive and false negative rates are computed as:

$$TPR = \frac{TP}{TP + FN} \tag{2.5}$$

$$FNR = \frac{TN}{TN + FP} \tag{2.6}$$

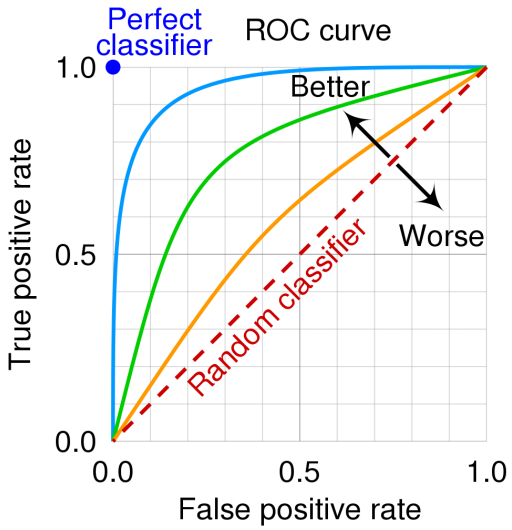


Figure 2.12: ROC curve [47].

The anomaly score determines how significant an abnormality is in comparison to other anomalies already seen. It is computed as the Mean Squared Error (MSE):

$$A_\theta(x) = \frac{1}{T}\sum_{t=1}^{T} \|\phi_t - D_{\theta_D}(\varepsilon_{\theta_E}(\phi_t))\|_2^2 \tag{2.7}$$

where $\|.\|_2$ is the l_2 norm, ε and D are the encoder and decoder functions, respectively, of the autoencoder, ϕ_t is the original input and T is the number of time-frames.

The way to compute the Area Under the Curve (AUC) and the partial Area Under the Curve (pAUC) were defined in the challenge as:

$$AUC = \frac{1}{N_- N_+}\sum_{i=1}^{N_-}\sum_{j=1}^{N_+} H\left(A_\theta\left(x_j^+\right) - A_\theta\left(x_i^-\right)\right) \tag{2.8}$$

$$pAUC = \frac{1}{\lfloor pN_- \rfloor N_+}\sum_{i=1}^{\lfloor N_- \rfloor}\sum_{j=1}^{N_+} H\left(A_\theta\left(x_j^+\right) - A_\theta\left(x_i^-\right)\right) \tag{2.9}$$

In these equations, $\lfloor . \rfloor$ is the flooring function and $H(x)$ returns 1 when the x is bigger than 0 and 0 otherwise. Here, N_- is the number of normal samples and N_+ is the number of anomalous samples. At last, $\{x_i^-\}_{i=1}^{N_-}$ and $\{x_j^+\}_{j=1}^{N_+}$ are the normal and anomalous samples, respectively. This samples are arranged in descending order by their anomaly scores. Thus, the threshold is set using the anomaly scores of normal test samples. This metrics are focused in binary classifications, whether the event exists or not, that is the approach taken for the anomaly event detector. pAUC is computed as the AUC over a low FP rate with a range from 0 to p, being p 0.1. This upper limit is defined that low since an anomaly detector cannot give false alerts constantly, it cannot be trusted.

However, a different way to do it would have been, instead of saying yes/no to anomaly identification, classifying the events and then establishing which of them is considered an anomaly. This approach would have used different metrics, precisely those used by the second system studied:

Precision: It measures the evaluation's of the algorithm model quality in classification tasks. In the area of event detection would refer to, out of all the events classified as X, how many were actually X. It is calculated as:

$$FNR = \frac{TP}{TP + FP} \tag{2.10}$$

Recall: It measures the percentage of affirmative cases that the algorithm accurately identifies. So, it basically shows, out of all the events that actually are X, how many have been correctly detected as event X. It can be computed as follows:

$$FNR = \frac{TP}{TP + FN} \tag{2.11}$$

F score: this metric combines the previous two in a single indicator. The greater this factor is, the better is the network performing. It is computed by:

$$Fscore = 2\frac{Precision\ Recall}{Precision + Recall} \tag{2.12}$$

Chapter 3

Implementation and Results

3.1 Introduction

In this chapter, the networks chosen for the development of the study and the experiments performed are going to be explained.

In the first phase of the project, the objective was to make an inventory of different networks designed to detect anomaly events from acoustic information and networks capable of classifying different acoustic events. Many where found, but the final selection was made based mainly in the libraries and code language used by the team who developed them.

Three experiments with the anomaly detector will be performed:

- Try to reproduce the results obtained by the developers of the network, preparing a virtual environment with the libraries needed and solving incompatibilities issues.
- Train the network with a new data base, created by joining two different data-sets suitable for the task of detecting anomalies in human environments.
- Modify main parameters of its structure in an attempt to improve the results obtained with the new data-set, trying to obtain the best architecture for the task.

Moreover, the second system selected, an events classifier, is tested to reproduce the results presented by its developer team. For so, a data base used for task of detecting real-life sound events, close to the aim of this project of detecting anomalous events in human life environments.

Throughout the chapter, GPUs are mentioned due to a few complications. GPUs are needed in tasks of Deep Learning (DL) since they allow parallel computations, exploiting matrix computation and obtaining that high computational capacity that CPUs lack. In this project, the computer running the network's codes uses an NVIDIA graphic cards, so the drivers Cuda and CuDNN are installed.

3.2 Anomaly Event Detection Network

The anomaly detector system initially used had been developed by one the teams participating, the *Pilastri_CCG* team, in the Detection and Classification of Acoustic Scenes and Events (DCASE) 2020 challenge. Specifically, for the Task 2: *Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring* [37]. It was chosen as the best approach since it was written in Python and it made use

of the libraries established as requirements (Keras and TensorFlow). It also ranked in a good position in the challenge.

Their approach is to develop a *Dense Autoencoder* which uses the sound energy features extracted from logarithmic mel-spectograms. They implemented an encoder of 4 fully-connected layers with 512 hidden units and a decoder of another 4 fully-connected layers, followed by Batch Normalization an ReLU as regularization technique and activation function, respectively. They also used early stopping of training to reduce the possible overfitting. As connection between both architectures (encoder and decoder), there is a latent space composed by a fully-connected layer with 8 hidden units. The structure can be easily identified in Figure 3.1.

Figure 3.1: Autoencoder network structure [39].

For training and evaluating the dense autoencoder, a database is provided by the DCASE challenge organizers. It is made by a selection of audios taken from the *ToyADMOS* [48] and the *MIMII Dataset* [49] databases. The resulting one consists of normal and anomalous operating sounds of 6 different toy and/or real machinery classes: Toy car, Toy conveyor, Valve, Pump, Fan and Slide rail. Each machine class has 4 different machine audios, except for the Toy conveyor which only consists of 3 different ones. All audios have a duration of approximately 10 seconds and only the information of one microphone in the microphone array was used, so that the audios are monoaural.

Since it is an unsupervised task, the data-set does not contain any labels. However, in the name of each audio track, it is specified whether it contains an anomalous event or a normal one. Thus, the format of every audio file is: `{normal/anomaly}_id_{idNumber}_{audioNumber}.wav`. Here, the first field indicates the normality nature of the file, the `idNumber` field is the identifier of every machine, for each class, and the `audioNumber` is just an extra identifier, but does not provide any relevant information.

For training, only normal audios are given to the network, while for evaluation a combination of normal and anomalous audios is used. The network creates a model for each machine class during training, minimizing the reconstruction error for every machine class' normal events. As explained in subsection 2.3.2, autoencoders compare the reconstruction error to a threshold to diffferentiate between anomaly and non-anomaly. In this case, as seen in equations 2.8 and 2.9, the threshold used to determine if there is anomaly or not in a file is the anomaly score (equation 2.7) of the normal test samples.

Three experiments are carried out with this network:

1. Reproduce the results presented in the DCASE challenge by the development team, with the objective of evaluating whether we were able to correctly reproduce it and the required environment conditions (mainly libraries and their versions).

2. Create a new data-set more adapted to our target scenario, and see how the network performs with it.

3. Change some of the network parameters to check their impact in the network performance.

3.2.1 Initial Experiments with the Original Network and Dataset

3.2.1.1 System Setup

The first experiment performed is to reproduce the results obtained by the original team just by running the autoencoder with the same data-set. There were problems derived from different library versions, incompatibilities with other libraries and different python versions.

First of all, it is needed to prepare a virtual environment, configuring the libraries' versions needed for running the network. For so, the Python version was updated to 3.7, since it is the one supported by Tensorflow 1.15 and the system makes use of this version of the library. A part from that, taking as reference a requirements file provided by the developers in their GitHub repository, several libraries are installed. However, a few incompatibilities errors appeared, so the developer team was contacted for an updated list of libraries and versions. Once the libraries are updated to the versions described in 2.4.1, so the incompatibilities are avoided between libraries and the code, the environment is ready.

Due to hardware limitation, a maximum number of samples of 2600 audio files is needed to be established. The GPU installed in the laboratory computer has less memory, so it can process less files during training than the one used by the developer team. This may affect to the results obtained, since there are less samples to create the models, as will be seen in the results discussion.

Once the virtual environment is prepared and the limitations are established, the system can be put to train and start to acquire results.

3.2.1.2 System Training

The network is trained with files from all the six machine classes and generates one model per each one of them. Each class has the following number of files for training the model:

- ToyCar: 4000 audio files.
- ToyConveyor: 3000 audio files.
- Fan: 3675 audio files.
- Pump: 3349 audio files.
- Slider: 2804 audio files.
- Valve: 3291 audio files.

This system is trained to minimize the Mean Squared Error (MSE), defined by equation 2.7, using the structure explained above and the Adam optimizer with a learning rate of 0.001. It also has implemented an early stop of the training so, if the error does not improve for 10 epochs, it stops training and saves the last best model obtained. This training method is iterated with a maximum of 100 epochs. It uses the 30% of the training samples in each machine as validation split.

Due to the hardware limitation mentioned, instead of taking the full set of files of every machine class, only 2600 audio files were taken for each one, making it a total of 15600 audio files. When training, it takes around 40 minutes to train each model, about 2 minutes per epoch, taking about 4 hours to perform the complete training of this data base.

3.2.1.3 System Evaluation and Results Discussion

For evaluating, the following amount of file are used:

- ToyCar: 2500 audio files.
- ToyConveyor: 3510 audio files.
- Fan: 1965 audio files.
- Pump: 856 audio files.
- Slider: 1290 audio files.
- Valve: 880 audio files.

It takes about 3 minutes to evaluate each machine class, taking around 18 minutes to finish and give the results. In Table 3.1, we show the comparison of the results presented by the original team and ours.

Machine type	**id**	**Project results**		**Developers results**	
		AUC	**pAUC**	**AUC**	**pAUC**
Fan	00	54.24%	49.53%	56.73%	49.72%
	02	73.06%	53.85%	79.60%	54.00%
	04	59.87%	52.25%	70.11%	54.11%
	06	73.86%	53.13%	81.69%	55.15%
Average		**65.26%**	**52.19%**	**72.03%**	**53.25%**
Pump	00	68.54%	55.58%	66.94%	56.83%
	02	61.95%	59.46%	60377%	60.31%
	04	84.03%	61.05%	57.00%	66.32%
	06	76.36%	59.86%	77.53%	60.32%
Average		**72.72%**	**58.99%**	**73.06%**	**60.95%**
Slider	00	97.58%	88.65%	96.12%	82.30%
	02	80.54%	64.12%	79.55%	64.42%
	04	93.49%	68.24%	95.44%	76.14%
	06	77.24%	49.62%	77.22%	49.56%
Average		**87.21%**	**67.66%**	**86.88%**	**68.11%**
Toy car	01	80.38%	69.21%	83.87%	72.64%
	02	84.72%	78.52%	87.56%	80.35%
	03	60.71%	54.39%	63.12%	55.02%
	04	82.84%	69.51%	88.60%	76.68%
Average		**77.16%**	**69.41%**	**80.79%**	**71.17%**
Toy conveyor	01	81.87%	68.40%	81.67%	69.41%
	02	66.52%	57.38%	68.04%	58.31%
	03	78.94%	62.19%	79.59%	63.64%
Average		**75.78%**	**62.66%**	**76.43%**	**63.79%**
Valve	00	74.51%	51.88%	74.61%	52.28%
	02	74.36%	51.80%	76.68%	52.72%
	04	77.86%	50.79%	79.58%	50.96%
	06	60.65%	48.25%	57.78%	48.73%
Average		**71.94%**	**50.68%**	**72.16%**	**51.17%**

Table 3.1: Results obtained in the first experiment.

The first thing observed is that in all the cases but one, our results are slightly worse than those of the developers. That is an expected result since the number of training samples was limited to 2600, due to that lack of memory space discussed before, so the network had less samples to create each model. Furthermore, the difference is not that big and the results can still be considered good enough, actually being even better than many results presented to the challenge task.

However, as mentioned above, this does not apply to slider machines. For all of the sliders machines, except for the one with id number 04, we obtained results that were better than those obtained by the developer team. This means that the original network was generating a model for slide machines too fitted to the training data. This leads to the model not correctly identifying anomalies in the evaluation test, since it is overfitted to the training data.

Our main conclusion, is that the system setup and adaptation we made was good enough to be considered valid, specially considering the limitations in the training data size. Moreover, we were able to identify a possibly overfitted result, which give interesting information and it raises the question of how to prevent it from happening.

3.2.2 Testing with a Different Database

3.2.2.1 System Setup

For the execution of this second experiment, the idea is to find several data-sets and create a new database by mixing them, so the adaptability of the autoencoder can be tested with it. The main target is evolving from a machine anomaly detection scenario to another more related to anomalies that can be found in human environments.

Consequently, an extensive research was carried out to find suitable databases, looking for those that could have both normal and anomalous events. In the end, as explained in section 2.4.3, two databases were selected for this purpose. Both are databases used in the DCASE 20017 challenge and were recorded with similar conditions of sampling frequency and sample resolution. From them, a differentiation between what is considered normal and anomalous events. Thus the events Gunshot, Glass breaking, Baby crying and Breaks squeaking have been defined as the anomalous events in the different scenes.

In both datasets, the audio files are composed by recordings of 3-5 minutes of duration. This is important since the autoencoder network under test was designed to work with files which last 10 seconds. Only the files with a duration shorter than 10 seconds were failing in both training and evaluating when taken to be introduced and processed by the network, resulting in void elements of a tuple. This problem came from an inaccuracy in the description given in documentation of the DCASE 2020 challenge, where it was said that the audios had an *approximate* duration of 10 seconds, when in fact their duration was *always* 10 seconds and the network was constructed only for 10-second files.

Then, before running the training, it was necessary to divide the audio files in 10-second segments. Moreover, it was needed to take into consideration that both anomalous and normal events were together in those long files. The first step was to extract all the anomalous events, that are those that really needed to be isolated. For doing so, the `sox` application was used, which allows to process audio files from the command line. With it, a few bash scripts were developed to process both databases. By using the provided labels (event name and starting and finishing timestamps) we isolated the anomalous events as individual audio files.

Since these events usually lasted less than 10 seconds, we decided to apply an strategy similar to *random-padding* [50] as opposed to performing a simple *zero-padding* approach. Random padding concatenates the same anomalous event file until it reaches the required 10 seconds duration. A better

approach would have been to modify the network's code to allow processing files with arbitrary durations, but for simplicity and due to a lack of time, this is proposed for future work.

Once the anomalous events were isolated, the files were divided into fixed 10 second segments, again developing several bash scripts and using the `sox` application. There was no need to keep the normal events separated in individual files since the network is a binary anomaly classifier.

Following the same format than the original database, the files were named differentiating between both data-sets. For doing so, the files obtained after processing the TUT Rare Sound Events 2017 data-set were given the id 00 and the ones obtained after processing the TUT Sound Events 2017 data base were given the id 01. This was made so that the network could differentiate results for the different datasets, even though what finally counts is the weighted average between both.

Once all this file processing was completed, the resulting database was composed of 1593 files for training and 648 for evaluation (489 files with id 00 and 159 with id 01).

3.2.2.2 System Training

Once the whole new data base was prepared, the network was trained to generate a unique model for this mixed data-set. In this experiment, the required time for the training was drastically reduced, since only 1593 files were processed, taking similar time per epoch than what it took in the previous experiment, around 1 minute and 50 seconds, and a completion time of 40 minutes more or less. The configuration of the training is exactly the same used for the previous experiment, except for the data-set used.

3.2.2.3 System Evaluation and Results discussion

As explained above, 648 audio files were used for the evaluation test, taking around 5 minutes to finish it and yield the results shown in Table 3.2.

Mixed dataset	**id**	**Project results**	
		AUC	**pAUC**
Gunshot, Baby crying Glass breaking	00	94.20%	85.85%
Brakes squeaking	01	69.81%	60.13%
Average		**88.21%**	**79.54%**

Table 3.2: Results obtained when using the new dataset.

These results are pretty good, as it can be seen that the weighted average classification rate is over 88% for the AUC metric, it is better than any result obtained with the original database used. There is a possible explanation for that. When listening to the different audio files generated, there are obvious differences between those containing anomalies and those that are normal events. In the anomalous files, all of them contain strong and strident noises. On the other hand the normal files contain very soft audios, clearly recorded in streets with very low traffic of vehicles and people, or a quite domestic environments, so when it appears a gunshot or a baby starts crying, it makes a noticeable difference.

Further analyzing the resultsl, the events of breaks squeaking obtained a lower hit rate, even though it could also be considered as a good result. That can be explained too by listening to the audios containing *Breaks squeaking* events. When expecting this kind of events, you imagine a deafening squeak, but it is just the sound of a vehicle stopping, not those high sounds expected. Nevertheless, since the network normalizes the input files before using them for training, it was able to correctly classify them as anomalous in a great number of cases.

3.2.3 Changing Network Parameters

3.2.3.1 System Setup

For the last experiment made with this network, the main parameters of the structure of the autoencoder were modified to see how the performance varied.

In this experiment, the dataset used was that composed during the completion of the second experiment (described in Section 3.2.2).

After discussing which parameters could be modified and tested, it was decided to perform the following modifications:

- Change the regularization technique from the Batch normalization used to a Dropout one with rates of 0.25, 0.5 and 0.75 (3 experiments, one per dropout rate).
- Change the number of neurons in every hidden layer to 128, 256 and 512 (original amount of hidden units).
- Modify the activation function from a ReLU to a Leaky ReLU and a PReLU.

The experiments were done independently, changing one parameter at a time, obtaining the results discussed below.

3.2.3.2 System Training

For this last experiment, it is used the data-set created for the previous experiment, thus having the same number of files for training and taking similar times for it. The main difference when training is that, every time a parameter is changed, it is needed to perform a new training, since the model created varies depending on the parameters selected and its values. Again, the configuration of the training is the same as that used for experiment one.

3.2.3.3 System Evaluation and Results Discussion

The first parameter to be modified and studied was the Dropout technique, instead of the Batch normalization. The results obtained are shown in Table 3.3.

Dropout	**id**	**Coefficient: 0.25**		**Coefficient: 0.50**		**Coefficient: 0.75**		**Batch normalization**	
		AUC	**pAUC**	**AUC**	**pAUC**	**AUC**	**pAUC**	**AUC**	**pAUC**
Gunshot, Baby crying Glass breaking	00	32.03%	47.83%	34.14%	47.86%	50.78%	48.05%	94.20%	85.85%
Brakes squeaking	01	64.66%	57.30%	57.30%	58.09%	32.78%	50.85%	69.81%%	60.13%
Average		**40.03%**	**49.70%**	**39.82%**	**50.37%**	**46.31%**	**48.73%**	**88.21%**	**79.54%**

Table 3.3: Results obtained when using Dropout.

A it can clearly be seen, there is an important worsening of the results, specially for the anomalous events taken from the TUT Rare Sound data-set (id: 00). However, what can be observed is the fact that, when increasing the dropout rate, the anomaly event detection is improved substantially, increasing the classification from a 32% to a 50% from the dropout rate of 0.25 to the 0.75 one. That makes sense since the dropout rate indicates the probability with which every neuron may or may not be stopped from learning and giving outputs, thus demonstrating the hypothesis of some studies that say the best

dropout rates are between 0.5 and 0.8. On the other hand, with the *Brakes squeaking* happens exactly the opposite, the network generates a good model for the smallest dropout rate, obtaining a result not that far from the original configuration, and it gets worsen when the dropout rate is increased. However, taking into consideration the weighted average results between both anomalous events, it still gets better when increasing the rate.

It can be concluded that the dropout approach is not adequate for this network.

The second modification is reducing the number of neurons per hidden layer to 128 and 256, in order to compare it with the results obtained when using 512 neurons. The batch normalization technique is again used. The results obtained are shown in Table 3.4.

Neurons	**id**	**128 neurons**		**256 neurons**		**512 neurons**	
		AUC	**pAUC**	**AUC**	**pAUC**	**AUC**	**pAUC**
Gunshot, Baby crying Glass breaking	00	91.38%	81.19%	92.09%	81.99%	94.20%	85.85%
Brakes squeaking	01	70.25%	61.29%	74.00%	61.11%	69.80%	60.13%
Average		**86.20%**	**76.31%**	**87.65%**	**76.86%**	**88.21%**	**79.54%**

Table 3.4: Results obtained when reducing the number of neurons.

According to the results, it is clear that reducing the number of neurons does not have a major impact in the network performance. The weighted average results are quite stable and always above 86%, which is a good success rate. When analyzing individually both anomalous sources, the *Brakes squeaking* events see an improvement in the performance of the network and the worst results are obtained when using 512 neurons per layer. It suggests that for this event, the network may be overfitting.

So, from this experiment, it can be concluded that is better to have 512 hidden units per layer, as the system has a better average hit rate. Future work should address further increasing the number of hidden units per layer.

The last modification done was to change the activation function of the autoencoder layers, by consiering the Leaky ReLU and the PReLU activation functions. The objective is comparing the original execution when using a ReLU activation function, leaving back the 512 neurons per layer. The results obtained are shown in Table 3.5

Activation funtions	**id**	**Leaky ReLU**		**PReLU**		**ReLU**	
		AUC	**pAUC**	**AUC**	**pAUC**	**AUC**	**pAUC**
Gunshot, Baby crying Glass breaking	00	90.07%	77.46%	93.65%	85.56%	94.20%	85.85%
Brakes squeaking	01	73.58%	62.50%	68.20%	59.95%	69.80%	60.13%
Average		**86.02%**	**73.79%**	**87.41%**	**79.27%**	**88.21%**	**79.54%**

Table 3.5: Results obtained when changing the activation funtion used.

Similar effect to those seen in the previous experiment can be observed. The results does not have a great variation and still can be considered good with all three activation functions, getting the worst results with the Leaky ReLU function, a bit better when using the PReLU function and the best results obtained with the original ReLU function. If we look closer, it can be commented that, while the *Gunshot, Baby crying, Glass breaking* events get better with every test, obtaining the best result with the original configuration, the *Brakes squeaking* event generates the best results when using the Leaky ReLU. For these events, the best model generated is when having a small leak in the activation function for negative values.

Thus, we finally conclude that the best activation function for this network, using this specific data-set, is the ReLU function.

3.3 Event classification network

3.3.1 System Setup

As explained in Section 2.4.3, we also wanted to evaluate the uses of an acoustic event classification network.

We initially decided to use the SELDnet network, developed by Sharath Adavanne and his team for the Task 3 of DCASE 2017 challenge: *Sound Event detection in real life audios* [46]. However, when planning how to isolate the detection part and ignore the location determination component of the system, we decided to finally use a simpler alternative, also available from the same team, which was already focused only in the event detection task. Thus, the SELDnet was discarded and SEDnet [10] was selected instead. This network was presented for the task of the DCASE 2017 challenge *Sound Event Detection in Real Life Audio*.

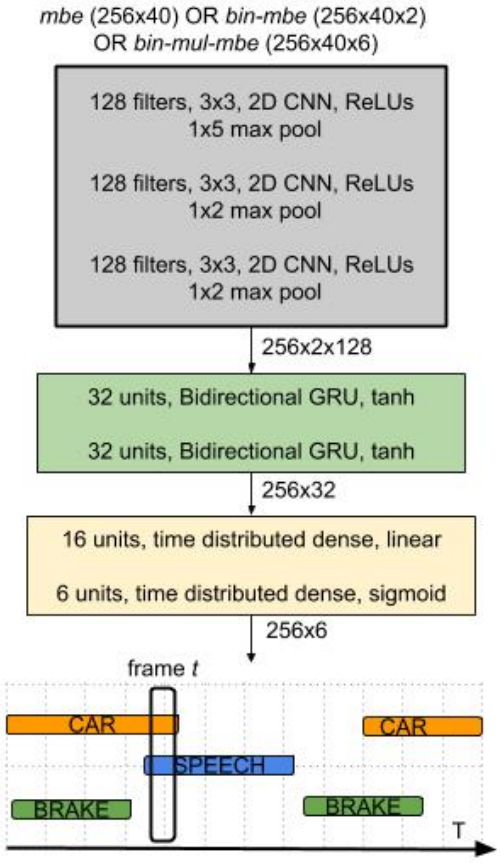


Figure 3.2: SEDnet network structure [10].

The architecture proposed is a Convolutional Recurrent Neural Network (CRNN) that takes the log mel-band energy feature, extracted from each channel of the audio input. Those features are supplied into the network, which maps them to the activities of the data-set sound event classes. For each of them, the neural network's output is in the continuous range of [0, 1], and corresponds to the likelihood

of that sound class being active in the frame. The ultimate binary determination of whether the sound event class is active or absent in each frame is obtained by thresholding this continuous range output.

Since is part of a DCASE challenge, the data-set used is given by the organizers [43], which is one of those explained and used in subsection 3.2.2, as part of the second experiment performed with the anomaly event detection network.

As stated in the objectives of this project, this network was studied aiming to reproduce the results obtained and presented by the team. There was no need for creating or updating a new environment for this network, it was enough with the configuration made for the anomaly detector.

3.3.2 System Evaluation and Results Discussion

Since the system uses a quite small database, the results present a considerable variation. For that reason, it is divided into 5 segments, using 4 in the training and the last for evaluating, making all the possible combinations. The system performance is calculated as the average of the results for all folds, which is presented in Table 3.6.

SEDnet	**Project results**		**Developers results**	
	Error rate	**F score**	**Error rate**	**F score**
Average results	58.69%	59.20%	60.00%	57.00%

Table 3.6: SEDnet results.

Both results are very similar, obtaining slightly better results in this project, since the error rate is lower (less events are misclassified) and greater F score. In both cases, the results are pretty good, and since the high variation of the results in every test, due to the reduced number of audio files, it can be considered the experiment of reproducing their results was successfully carried out.

It would have been of great interest to also obtain the confusion matrix metric, since it provides information about how each event is classified. However, the original code contained some bugs in its calculation and, after a few tries, it was not possible to find the exact problem. With more time available, it would have been solved and discussed.

Chapter 4

Conclusions and future work

4.1 Conclusions

Throughout this project, several objectives were pursued and conclusions were derived after every phase of it, allowing the better understanding of deep learning and its tools, applied in an acoustic anomaly detection scenario.

The first step was to make a general review of concepts related to the topic so it was possible to carry out the studies and experiments proposed. In Chapter 2, an explanation of the main concepts needed for the correct understanding of the project was given, to allow for an easier understand of this document and the experiments and results described in it. Additionally, an inventory of available systems and data-sets was described, defining the pros a cons of every system and data-set found, giving an explanation of the criteria used to select the detection systems and databases. With all this context, the first goals of the project were achieved, having a good idea about the deep learning technology, and the various networks and data-sets to work with.

After the completion of these goals, we selected two systems available in the scientific community, for the anomaly detection and event detection taks. from them, we generated working systems and conducted several experiments with them. Thus the anomaly event detection system was prepared to run with both the original data-set and a new one fully designed by combining the content of publicly available databases. Additionally, we introduced and evaluated some modifications to the system architecture and the available control parameters. These implementations, modifications and the obtained results are described and discussed in Chapter 3. Finally, we also addressed an initial evaluation of the selected acoustic event detection system, preparing a computer environment so it could be run and tested, comparing the results obtained with those given by the team that created it.

The main objective of the project was to carry out the study, implementation and evaluation of systems capable of detecting events, focusing in the detection of anomalies, using only acoustic information. With the project completed and all the results discussed, we believe that this objective was accomplished.

4.2 Future work

Due to a lack of time, not all the modifications and experiments could be fully developed and fully detailed in this document. Thus, as a possible future investigation, the dense autoencoder code could be deeply analyze to allow it to to process files with arbitrary lengths. Additional modifications of the

system architecture and control parameter could be done, with special focus in carrying out a much more detailed study on its evaluation, also addressing additional and more complex datasets.

Another research line could be carried out with the dense autoencoder, consisting in the combination of several parameter modifications, not only independent changes, so that the best architecture for the task of anomaly detection could be accomplished with this network.

In what respect to the acoustic event detection network used, it would be neccessary to fix the calculation of the confusion matrix, since it gives very interesting information that could not be obtained during the development of this project. Additionally, and as it has been done with the dense autoencoder, the experiments 2 and 3 could be conducted to study the adaptability of this network to face different data-sets containing additional sound events and modifying its architecture. The second experiment, in which a new data base was created and used, would be of high interest since the given one had a reduced number of samples and classes.

Finally, it would be interesting to take the last step and study systems for the acoustic scene classification task, as the natural next goal to achieve, since it consists on detecting many different events and classify the whole as a scene, for example the street, a basketball match or a domestic environment.

Bibliography

[1] G. Vallathan, A. John, C. S. Thirumalai, S. Mohan, G. Srivastava, and C.-W. Lin, "Suspicious activity detection using deep learning in secure assisted living iot environments," *The Journal of Supercomputing*, vol. 77, 04 2021.

[2] J. Xie and M. Zhu, "Investigation of acoustic and visual features for acoustic scene classification," *Expert Systems with Applications*, vol. 126, pp. 20 – 29, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417419300661

[3] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 379–393, 2018.

[4] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135.

[5] E. Saykol, M. Bastan, G. Ugur, and zgr Ulusoy, "Keyframe labeling technique for surveillance event classification," *Optical Engineering*, vol. 49, no. 11, pp. 1 – 12, 2010. [Online]. Available: https://doi.org/10.1117/1.3509270

[6] J. Feng, C. Zhang, and P. Hao, "Online learning with self-organizing maps for anomaly detection in crowd scenes," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3599–3602.

[7] W. Zajdel, J. D. Krijnders, T. Andringa, and D. M. Gavrila, "Cassandra: audio-video sensor fusion for aggression detection," in *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, 2007, pp. 200–205.

[8] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor, "Resnetcrowd: A residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–7.

[9] D. Fuentes, "Deep learning e inteligencia artificial con keras/tensorflow," https://www.udemy.com/course/curso-de-deep-learning-con-kerastensorflow-en-python, 2014.

[10] S. Adavanne, P. Pertilä, and T. Virtanen, "Sound event detection using spatial features and convolutional recurrent neural network," 2017.

[11] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Processing Magazine*, vol. 38, no. 5, pp. 67–83, Sep 2021. [Online]. Available: http://dx.doi.org/10.1109/MSP.2021.3090678

[12] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: A systematic review," *ACM Comput. Surv.*, vol. 48, no. 4, Feb. 2016. [Online]. Available: https://doi.org/10.1145/2871183

[13] A. Harma, M. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *2005 IEEE International Conference on Multimedia and Expo*, 2005.

[14] Chrislb. (sep) Artificial neural network. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network

[15] S. Pyo, J. Lee, M. Cha, and H. Jang, "Predictability of machine learning techniques to forecast the trends of market index prices: Hypothesis testing for the korean stock markets," *PLOS ONE*, vol. 12, p. e0188107, 11 2017.

[16] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[17] J. O. Alvear, "Arboles de decision y random forest," nov 2018.

[18] Swapna. Convolution neural network. [Online]. Available: https://developersbreach.com/convolution-neural-network-deep-learning/

[19] Y. su and C. Kuo, "On extended long short-term memory and dependent bidirectional recurrent neural network," *Neurocomputing*, vol. 356, 02 2018.

[20] O. I. Provotar, Y. M. Linder, and M. M. Veres, "Unsupervised anomaly detection in time series using lstm-based autoencoders," in *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*. IEEE, 2019, pp. 513–517.

[21] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, and N. Harada, "Unsupervised detection of anomalous sound based on deep learning and the neyman–pearson lemma," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 212–224, 2018.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[23] F. Chollet. (2016, may) Building autoencoders in keras. [Online]. Available: https://blog.keras.io/building-autoencoders-in-keras.html

[24] J. Brownlee, "How to choose an activation function for deep learning," Jan 2021. [Online]. Available: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

[25] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2019.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[27] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Citeseer, 2013, p. 3.

[28] S. Nandi. (2021, aug) Facial expression recognition & comparative study on densenet161 and resnet152 using deep learning, pytorch, and transfer learning. [Online]. Available: https://medium.com/analytics-vidhya/facial-expression-recognition-comparative-study-on-densenet161-and-resnet152-e88ee02b6734

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[31] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/076a0c97d09cf1a0ec3e19c7f2529f2b-Paper.pdf

[32] E. Cakır and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," *Deep Neural Networks for Sound Event Detection*, vol. 12, 2019.

[33] Q. Gao, Z. Li, and J. Pan, "A convolutional neural network for airport security inspection of dangerous goods," *IOP Conference Series: Earth and Environmental Science*, vol. 252, p. 042042, 07 2019.

[34] "Python programming language webpage," http://python.org [Last access 3/November/2020].

[35] F. Chollet and others, "Keras: The Python Deep Learning library," p. ascl:1806.022, Jun. 2018.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016.

[37] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, November 2020, pp. 81–85. [Online]. Available: http://dcase.community/documents/workshop2020/proceedings/DCASE2020Workshop_Koizumi_3.pdf

[38] D. Lapin, M. Khubbatulin, and V. Klychnikov, "The study of anomalous machine sound detection based on cyclostationarity model," DCASE2020 Challenge, Tech. Rep., July 2020.

[39] A. Ribeiro, L. M. Matos, P. J. Pereira, E. C. Nunes, A. L. Ferreira, P. Cortez, and A. Pilastri, "Deep dense and convolutional autoencoders for unsupervised anomaly detection in machine condition sounds," 2020.

[40] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 1, pp. 34–48, Mar 2019. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2018.2885636

[41] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: an open dataset of human-labeled sound events," *arXiv preprint arXiv:2010.00475*, 2020.

[42] M. Cartwright, J. Cramer, A. E. M. Mendez, Y. Wang, H.-H. Wu, V. Lostanlen, M. Fuentes, G. Dove, C. Mydlarz, J. Salamon *et al.*, "Sonyc-ust-v2: An urban sound tagging dataset with spatiotemporal context," *arXiv preprint arXiv:2009.05188*, 2020.

[43] A. Mesaros, T. Heittola, and V. Tuomas, "TUT database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.

[44] A. Mesaros, T. Heittola, and T. Virtanen, "TUT sound events 2017, development dataset." Zenodo, mar 2017. [Online]. Available: https://doi.org/10.5281/zenodo.814831

[45] ——, "Acoustic scene classification: An overview of DCASE 2017 challenge entries," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, September 2018, pp. 411–415.

[46] A. Mesaros, A. Diment, B. Elizalde, T. Heittola, E. Vincent, B. Raj, and T. Virtanen, "Sound event detection in the DCASE 2017 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019, in press.

[47] MartinThoma. (2018, jun) Artificial neural network. [Online]. Available: https://en.wikipedia.org/wiki/Receiver_operating_characteristic#/media/File:Roc_curve.svg

[48] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, November 2019, pp. 308–312. [Online]. Available: https://ieeexplore.ieee.org/document/8937164

[49] H. Purohit, R. Tanabe, T. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, November 2019, pp. 209–213. [Online]. Available: http://dcase.community/documents/workshop2019/proceedings/DCASE2019Workshop_Purohit_21.pdf

[50] X. Dong, B. Yin, Y. Cong, Z. Du, and X. Huang, "Environment sound event classification with a two-stream convolutional neural network," *IEEE Access*, vol. 8, pp. 125 714–125 721, 2020.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá