uc3m | Universidad Carlos III de Madrid

University Degree in Audiovisual Systems
Academic Year (e.g. 2014-2019)

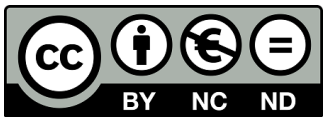
Bachelor Thesis

"Violent event detection from acoustic signals"

Óscar Otero Martínez

Carmen Peláez Moreno

Madrid, February 18, 2020



SUMMARY

Keywords:

DEDICATION

CONTENTS

CPM: algo raro pasa con esta tabla de contenidos porque la bibliografía aparece en

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Context

Violence against women remains an invisible phenomenon, deeply within the victim's private life in most cases. It is based on deep social and cultural roots and it is undoubtedly linked to unbalanced relationships between men and women in different situations and contexts, such as economics, politics and religion. In order to prevent these conflicts, the related legislation has achieved important improvements for the last years. According to the results of most of the studies, victims can be usually defined as women who endured violence during their childhood and felt socially isolated. They are also characterized for a considerable economic dependency and a low educational level.

With the purpose of making a difference when identifying situations showing this kind of violence and applying the knowledge and technological advances acquired during this information era, machine learning and deep learning models can collect all the available data to protect eventual victims.

The main challenge is to get to know how the victim is feeling when she is under a threatening situation, for example, if she is scared or nervous, and combine this with other variables which may play an important role in the scene and might be helpful in making a decision about its characterization. There are several factors that can be considered to achieve this task. One of them is the audio, either the victim's voice or the environmental sounds.

Plenty of useful information can be extracted from the acoustic scene. The detection of audio events can be combined to understand what is happening in a certain moment. Once these data are collected, they can be classified in different categories and thus describe several aspects of the acoustic scene. Either based on an objective definition of gender violence or on an explanation previously obtained from a particular/specific victim, this acoustic knowledge can be interpreted as dangerous for the user.

1.2. Objectives

in a beginning, the main objective of the work was to design and implement a system that allowed to detect and identify violent events from acoustic signals. In order to define a way of how to address the problem, one of the idea was to work in a sound detector. However, how do we know exactly what kind of sounds do we want to detect? Then, we chose the path of defining a system based on a classifier of acoustic events that gives as final output a binary tag identifying the input act as violent or non-violent to improve on the definition of violence. This is basically the global objective of the whole project. The

CPM: falta la cita.

CPM: tienes puesto que vas a introducir unas refs. Es importante porque no se pueden mencionar estudios de esta manera "most of the studies" sin citar algunos al menos

following list collects the different partial objectives considered necessary to obtain the final one.

- To find a data resource, for example, a public database, that counts with a sufficient number of data observations, especially, that belong to a certain type of classes that can be considered as violent in different situations.
- To discover a set of features or a feature space in which it is feasible to obtain a more clear separation between violent and non-violent events than the one obtain with more common feature extraction methods.
- To define an idea of violence that fully adapt to the victim situation which hast not to be necessary considered as violence from a more objective point of view.
- To implement a system that allows the user to transmit the whole model her own definition of violence so it can get adapted to the victim's personal situation.
- To find an algorithm that allows to take advantage of the new features in order to perform a multiclass classification to differentiate between violent and non-violent events.
- To adapt the output obtained from the multiclass task of the model to a binary classification in order to finally identify violent actions.

1.3. Regulatory framework

Tips?

1.4. Socio-economic environment

CPM: por ejemplo, asuntos relacionados con la ley de protección de datos, la ley integral de violencia de género

2. STATE-OF-THE-ART

Since the main topic of this work, acoustic violent scene detection, has not been researched as such in the literature before, we decided to start our search by addressing close two topics that have been largely studied for the last years related to acoustics scenes and events also including some aspects on violence detection mainly aimed at the detection of violent scenes in movies.

2.1. Acoustic Scene Classification and Acoustic Event Detection and Classification

Acoustic Scene Classification (ASC) refers to the association of an audio sequence to a certain semantic label that describes the environment in which it took place [1]. With this idea in mind, the classification of acoustic scenarios has been tackled with two different kinds of concepts: soundscape cognition, i.e. understanding how the human being perceives the sounds subjectively from the physical environment that surrounds them [2], and Computational Acoustic Scenes Analysis (CASA), that is working on new computational methods that may help automatize this task through machine learning and processing signal techniques [3]. This notion can has many applications, such as content recognition –by allowing devices to obtain benefits and information from its situation [4]–, for medical usage [5], as a tool to aid musical recognition [6] or as a complement to Computer Vision (CV).

Simultaneously to these advances in the ASC, another related area has evolved during the last years: Acoustic Event Detection and Classification (AED/C). It can be described as the processing or treatment of sound signals in order to convert them into significant descriptions that match a listener's sensing of the events and sources composing the acoustic environment [7]. The detection part consists on identifying the events in a temporal stream of audio and labelling them. The result is usually accompanied by the time interval in which the occurrence is set. However, classification is a task that acts directly on the event that has been already isolated and has the purpose of designating a label or class to the sound [8]. These techniques also have plenty of applications, e.g., in the medical field [9], in biological topics such as bird noise detection [10], and for multimedia information retrieval from video sources in social media [11], etc.

2.2. Features and methods for ASC and AED/C

As in many fields, the boundaries between features and methods used for audio tasks are becoming blurred since the recent rise of deep learning based methods . Classically, a problem is addressed with a pre-processing stage of the data and then the model or method is implemented. Right now, these two stages are sometimes maintained but also

Adding reference to deep learning

have been mixed or changed depending on how the algorithm works.

In every machine learning or pattern recognition task, for the system to be able to infer and extract conclusions from the input given, a pre-processing stage is necessary to make some transformations to the data so that they could be understood by the model. This stage is known as feature extraction and the goal is to convert the original information into a set of values or vectors that characterize the data regarding some desired properties [12].

There are several ways that allow us to perform this processing stage. The most common and basic one consists on extracting features that are closely related to the original signal which are called low-level descriptors (LLD) [13]. These are computed by performing some mathematical operations or formulas to the original data and can be considered rudimentary when comparing with other techniques. However, they are really extended and still in use nowadays [14].

In the audio field, there are two types in which all LLD can be grouped into. One of them is for the features that have been computed by considering the audio signal in its original form in the recording, i.e., in time-domain, that is the reason why they are known as time-domain audio features. The other case refers to those characteristics that are obtained from the signal after having been transformed into the frequency domain. These are commonly known as frequency-domain or spectral audio features. For the procedure of feature extraction, the signal is usually divided into frames that can be overlapped by using a sliding window, so the calculations are done per frame, obtaining a final matrix with size $number\ of\ frames \times number\ of\ features$ [12]. It must be taken into account that the goal of the whole system is going to be fundamental at the time of deciding which features must be computed. For example, not the same features are to be used for speech recognition than for musical information retrieval. In table 2.1, a summary of most used time and spectral features is included.

Moreover, either time-domain or spectral features can also be split in two groups depending on the way the were extracted. There are the ones called short-term features. This way of computing follows the framing process explained above. The signal is initially divided in various frames that are commonly overlapped and for each frame the feature is calculated [12]. The other case refers to long-term features. This way the different computations to obtain the descriptors are performed along longer segments, sometimes the whole audio is took [17]. For a certain long sequence, the computation of the short term features are calculated and then some statistics are performed in order to characterize this information for the long term. Actually, the long-term sequence can be defined as a set of statistics that resumes the short-term features for a sequence. The portion of audio selected for the long-term usually presents a uniform behaviour within the total data [12].

To sum up, a typical procedure would first divide the signal into frames. The next step would be to compute the short-term features considering some time-domain or spectral characteristics within the audio data that are thought to be determinant in the required

Feature	Description
Time	
Energy Entropy	It is useful to detect sudden changes from the energy of a signal. To calculate this value for a certain subframe, it is necessary to first compute the normalized energy of the subframe with respect to all the frames energy [15].
Short time energy	It is the energy for a short segment of signal. It is normally used in speech-related tasks in order to identify voiced form non-voiced fragments [16], among other things.
Zero Crossing Rate (ZCR)	This can be defined as the number of times the amplitude of the signal crosses the zero line per unit of time, i.e., changes from negative to positive. It is sometimes computed by the number of zero-crossings by the amount of samples in the frame [15].
Frequency	
Spectral Flux (SF)	It is computed to measure the spectral changes between two successive frames. To do so, the difference of their spectral energy obtained from the FFT is computed [16].
Spectral Rolloff	This represents the skewness of the shape of the spectrum given the frequency below which a concrete percentage of the magnitude distribution of the frequency transform is concentrated [16].
Spectral Centroid (SC)	This is a measure related to the spectral position. It is defined as the center of gravity of the spectrum. [15].
MFCC	It is a feature that it is widely used because it gives good results in many tasks as for example, speech recognition since it interprets the frequency bands in a very similar way to human perception while doing a separation of the fine structure of the spectra (that corresponding with the harmonics) and the coarse (i.e. the filter representing the vocal tract). It is computed from the STFT [16]. A wider explanation can be found in appendix .4.

Table 2.1. EXAMPLES OF TIME AND SPECTRAL AUDIO FEATURES.

task. Finally, a series of mathematical and statistical transformations are performed in order to obtain a LLD vector per sample in order to pass these input data to a statistical model that will be trained with the objective of summarizing the properties of the signals

and develop a classification rule so as to be able to assign a faithful category to a new unlabelled observation [18]. For example, some of the long-term features that can be obtained from the short-term ones included in 2.1 could be the variance and mean in the case of short time energy, or the relation between the maximum and the mean along the frame for the SF [15].

A common approach for the utilization of this type of LLD in the long-term way is the one known as Bag-of-frames (BOF). The term has been acquired from the text treatment field as an analogy to the technique known as "bag-of-words", which consists on treating the text data as a global and unique distribution of words without giving importance to their order inside the sentences [19]. The analogous system for audio tasks consists in representing signals as statistics of long-term sequences of their spectral local features. This method has been applied in different fields in the ASC, for example, in order to analyze "soundscapes", analogy for audio landscape, and polyphonic music [20]. It was established that this approach was a proper option in order to work with soundscapes since the order of the audio sequence inside of them was not really relevant for its understanding. However, in the music task, the ordering of the parts of the sound was determinant to retrieve the musical information, so the technique was discarded for this topic [21]. A common way of exploiting this concept of BOF is by using the already mentioned MFCC and the really well-known technique in the literature as GMM. These are generative methods in which the feature vectors are treated as they were produced by a multimodal distribution which consists on a sum of Gaussian distributions [1] . In figure 2.1, it is shown the structure of a Bag-of-frames approach.

check

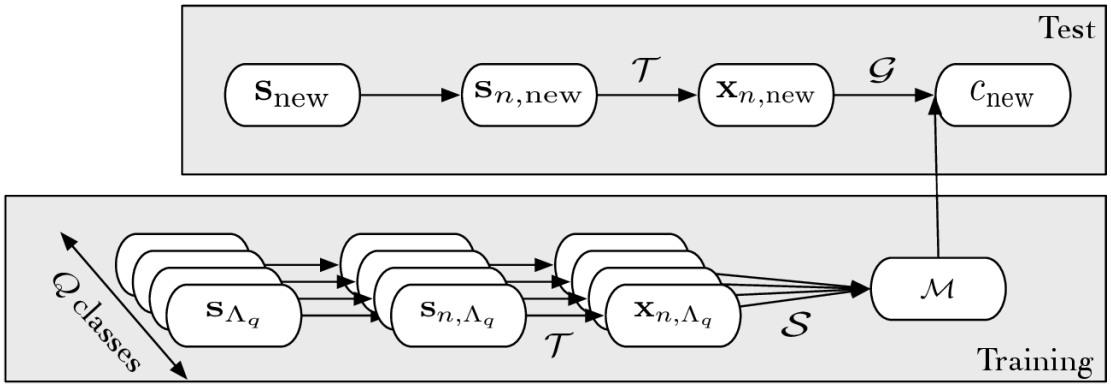


Fig. 2.1. Framework for ASC addressing the idea of BOF and using a GMM to obtain statistical representation of the low-level features. $S_{\Lambda q}$ are the original labelled samples in the training set and $S_{n,\Lambda q}$ multimodal distributions for these samples. $x_{n,\Lambda q}$ are the features extracted with an operator T which are taken by an operator S to learn the global model M. Then, in the testing process, a likelihood measure G is used in order to classify the *new* sample.

A similar way of acting can be found for AED. In this case, the typical procedure has been found to be defined as a two-steps process. For the case of Multimedia Event Detection (MED) [11], in the first part, some features are generated at a clip level, usually

by concatenating frame-level features, so then, in the second stage, a binary or multiclass classification can be done in order to confirm the presence of an event in a whole clip of video. This idea follows the same concept of BOF explained above since the generation of features do not attend to their order in the whole clip.

Another similar approach has been found for this task, but in this case it differs from the BOF and it is applied to real-life recordings. For the first stage, a classification of already isolated events is performed so a vocabulary of acoustic actions can be built. For this purpose, it is needed a set of short-term video recordings in which the semantic meaning of the corresponding audio event must be clear and highlighted over all the others sounds that take place along the sequence. In the second stage, the detection procedure takes place along the whole clip by making use of the previous configured vocabulary of acoustic data [22]. The way the events are modeled in this task is by using Hidden Markov Models (HMM). This is a common technique that has been put in practice plenty of times in the audio world and consists in modeling the temporal succession of the events within a longer sequence. This can be done by saving the order of the events in a transition matrix that contains the probability of one sound occurring after another one [1].

Also, this process can be combined with a classification of the events detected during the long sequence in order to make an ASC based on what type of sounds are taken place within the scene [1]. This is actually the initial idea desired for our work. Building a violent acoustic dictionary with a classification of monolabel short time recordings so as to then perform a detection process for violent acoustic events in order to categorize the scene and check if there is violence on it. However, we just mainly made an approach for the very first part about classifying what we considered violent events. This will be explained further on in 4 and 5.

In some cases, the mentioned low-level descriptors, sometimes referred as manually-crafted features, are not enough to model data since they cannot achieve a meaningful representation of the original information for the system to learn from. Also, as mentioned above, the selection process of this type of features plays a significant role in the final output so that the criteria of the designer could be considered a limitation for the classification task [23]. For this reason, a new strategy of addressing feature extraction has appeared based on the idea of finding more specialized properties by using specific engineering algorithms that explores the given data in order to find non-human recognizable patterns. The techniques that allow to perform this task are usually called automated feature generation algorithms or machine-learnt features [24].

In order to compute these high-level features, there is not just one standardized machine model that allow to compute them all in a particular form. The calculation method, nevertheless, differs from one approach to another. One example could be the combination of low-level data with high-level semantic descriptors that consists on the inference of diverse dimensions by using Support Vector Machines so as to find similarities among music genres. Particularly, they consider the output probabilities of the SVM classifier as

a high-level feature space in which the distance between samples from different classes can be measured [25]. This algorithm has been one of the most popular ways to track classification problems.

There are plenty of works in the literature that use this technique for audio classification tasks [26] [27] [1]. It is originally known to be a binary classifier but, nowadays, there have been some implementations that allow its use it in multiclass problems. This algorithm makes the classification by taking into account pairs of observations that are more likely to be misclassified and draws a hyperplane as the optimal boundary between the two classes [28]. Since it has been used for the experiments in this work, a more detailed explanation will be included in section 4.3.

Several methods that have been really successful on the high-level feature extraction apart from other tasks as detection and classification lay under the umbrella of Artificial Neural Network (ANN). This concept was inspired by human biology and how the neurons communicate among them in the brain in order to interpret the input or sensory data humans collect.

In the field of AED these have been used in order to implement techniques also for automatic learning of features. For example, a technique of boosting is used to extract discriminative features which resulted in a better performance than MFCC for real-life acoustic data [29]. They first made an attempt based on modeling the sequence data with HMM explained before and combined them with a trained ANN. Also, an approach based on SVM and GMM was tried.

The development of the ANN in different ways has made possible the implementation and creation of new algorithms during the last years that exploits the concept of neural models from different perspectives. Is the case of Convolutional Neural Network (CNN), which follows the way of working of the neural networks but performing convolution operations over the input data [28]. This has achieved really good results for image tasks in the Computer Vision field, but also implemented successfully in audio problems. As in [30], where is proposed a method based on CNN in order to classify ten acoustic scenes. In order to feed the model they used log-mel spectrogram images. More information about this type of data can be found in appendix .4. A detailed explanation of CNN can be found in subsection 3.2.1.

Also, another type of networks that has been treated largely with the purpose of working with sequential data is the Recurrent Neural Network (RNN). The concept behind these models is its capability of being able to learn information from previous elements inside a sequence. Also, an adaptive model was designed called LSTM, which do basically the same but for longer sequences. One possible approach for this algorithms could be the one presented in [11]. They first classify frame level audio against a set of semantic units. Then, the output is taken as a representation with variable length in a clip level which feed the LSTM as input. The results overtake SVM and FC in similar tasks. More information about this type of networks is included in detail in subsection 4.3.3.

Related to the development of these more complex models as CNN or LSTM, a novel type of feature has been used in the last years that differs from the already explained low-level and high-level ones: Deep Neural Network (DNN) have grown increasingly for plenty of classification tasks and so in the multimedia area. The problem with this type of systems is the huge amount of data that is needed to make them work properly, which can be translated in a lack of labelled data. One of the habits that has been currently resorted by the researchers consists of learning what is called deep data *embeddings* from extensive collections of, in our case, audio and use them so as to perform shallow classifications by using simpler datasets. There have been implemented some models about this topic, such as Look, Listen and Learn (L^3) [31] net that uses as input for the the embedding extractor the linear-frequency log-magnitude spectrogram of 60 million audio samples, the system called SoundNet [32] that has been designed to obtain embeddings from training a deep audio classifier in order to predict the output o a deep image classifier and the VGGish network, designed by Google researchers. This last case is the one we used in this work and it will be explained in more detail in section 3.2.

One of the approaches that can be found in the literature about this idea is based on the use of a previously learned over-complete dictionary that is utilized to sparsely decompose the spectrogram of audio. This dictionary will be used by an encoder with the purpose of mapping new input data to real similar versions of their own sparse representation in a fast and efficient way. Finally, the obtained codes will feed a Support Vector Machine (SVM) classifier, used for the task of music genre prediction [33].

Another job done in the sparse-feature representation framework presents a way of mixing high feature learning techniques with a pooling method for the objective of music information retrieval and annotation. After some preprocessing of the audio signals data, three feature-learning algorithms are trained finding that sparse restricted Boltzmann machine (sparse-RBM) gets better results than K-means and Sparse Coding. Once the features are obtained, an extra step takes place before performing the classification task, the one called pooling and aggregation. The goal of this procedure is to achieve a feature representation for a long sequence such as a song. Since when joining short-term features that belong to small segments inside the song may result in a loss of their local meaning, a max-pooling operation is computed over each subsegment in order to just consider the maximum value for each feature dimension. After that, these are aggregated by computing the average. The max-pooling contribution resides on reducing the smoothing effect when averaging the values [34]. This approach is feasible because of the homogeneity in music data. However, this technique could be slightly risky when dealing with acoustic scenes. For this case, a modified version of this method has been proposed. Taking into account that the presence of events is less frequent, instead of considering the whole long sequence to apply the max-pooling for, it will just be used in those segments that had been already detected as significant events by establishing a threshold value and setting an onset and offset that allow to know the start and end time [35]. **The representation of the audio event in a feature space explained in this case is the one that better fits our**

CPM: aquí hace falta que pongas ejemplos de feature extraction en AED. Basta un párrafo con dos o tres refs. Si no, no se sabe qué relación tiene esto con nuestro problema y

approach (4.2) until now

2.2.1. Transfer learning

All these deep models mentioned above in 2.2 allow us to use complex extractors trained with huge collections of data and apply them to models considerably much less complicated to address other type of problems. This is possible due to a kind of techniques commonly known as transfer learning and domain adaptation.

The typical consideration in plenty of machine learning tasks consists on extracting the training and testing subsets of data from the same feature space and same distribution. When one of these initial assumptions change, it is necessary to rebuild the whole model from the initial point, including new training data, which means a lot of computational cost and loss of efficiency [36]. Its working manner can be explained as an analogy of how humans transforms their ability on a certain task to obtain knowledge for other purpose. An example could be how musicians apply their previous experience to get to know faster how to play another instrument.

The first work in which this topic has been treated widely was in 1995 in the workshop *Learning to Learn* [37] and since then many approaches have arisen and renamed the same idea as knowledge consolidation or inductive transfer. However, it was 10 ten years later, in 2005, when the first idea of the ability of a system to identify and apply learned skills previously to completely new problems appeared from the hand of the Broad Agency Announcement (BAA) 05-29 of Defense Advanced Research Projects Agency (DARPA)'s Information Processing Technology Office (IPTO) [36]. This can be expressed as a relation between a *source* task, where the abilities are learned, and *target* task, the novel problem that needs to be resolved. As a difference with other similar methods, in this concept of transfer learning the roles of these two are not equal since the weight of the target is much heavier. In figure 2.2 it is shown the difference between a common machine learning approach and the use of transfer learning.

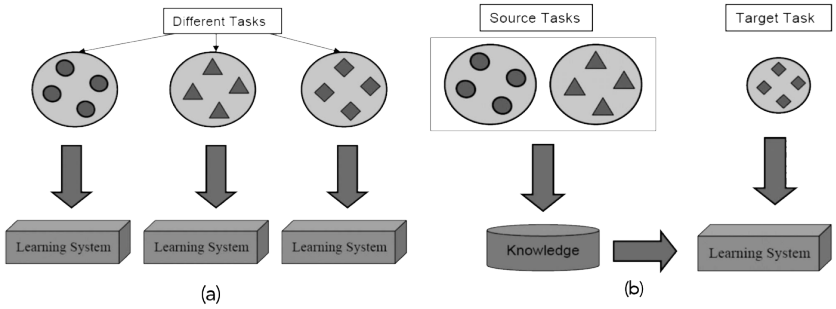


Fig. 2.2. Difference between traditional machine learning (a) process and feature learning (b) [36]

The same idea can be understood from a mathematical point of view as the analysis of the relationship between the two different spaces from types of targets [36].

Considering a *domain* D that is composed by a feature space denoted by X and a

Include picture of the pipeline? CPM: pero aquí no. Si lo haces, en el capítulo de Methods y aquí dices en qué sección lo desarrollas.

mantener estos parrafos

small intro to data augmentation?

marginal probability named $P(X)$, where $X = \{x_1, ..., x_n\} \in \chi$. The whole domain can be expressed as $D = \{\chi, P(X)\}$ where x_i is a certain vector inside the feature space.

In the same way, a *task* can be defined as T formed by a label space γ and an objective predictive function η. The task formulation is $T = \{\gamma, \eta\}\}$. This predictive function cannot be observed, however the intention is to learn it from the training data, that is composed by pairs of the form $\{x_i, y_i\}$, $x_i \in X$ and $y_i \in \gamma$.

The predictive function η can be used to predict a corresponding label of a new sample x. From a probabilistic perspective, this new label can be expressed as $P(y|x)$. So, the task T can be defined as $T = \{y, P(Y|X)\}$, in which $Y = \{y_1, ..., y_n\} \in \gamma$. For each vector x_i, the function η finds a prediction y_i.

Once these parameters have been defined, considering the source domain D_S, task of source domain T_s, the target domain as D_T and its respective task as T_T, the transfer learning has the purpose of obtain the condition distribution in the target domain $P(Y_T|X_T)$ with the information extracted from D_S and T_S where $D_S \neq D_T$ or $T_S \neq T_T$ [38].

We have took advantage of this technique in order to obtain features for our final model. As it is explained further on in 3.2.3, the network VGGish has been pretrained with a huge dataset with similar data as the one we take. Thanks to this method of transferring learning from models, we can extract our features without need of training the complex VGGish, which eases the whole process in a considerable way.

2.3. Violent Event Detection

All the multimedia information available can be applied to many fields and with different connotations. One of the variants that has appeared in the acoustic scenes and events field is the one applied to violence detection. For this case, an essential point before addressing any problem is to decide what kind of definition the word 'violence' is going to adopt since it is a subjective concept. An objective perspective has been given by the World Health Organization as "The intentional use of physical force or power, threatened or actual, against oneself, another person, or against a group or community, that either results in or has a high likelihood of resulting in injury, death, psychological harm, maldevelopment or deprivation" [39]. There are other definitions found in different works as "physical violence or accident resulting in human injury or pain" [40] or "any situation or action that may cause physical or mental harm to one or more persons" [15].

Recent studies have treated this problem in different ways due to all types of conditions that this may take place in. During the last years, the possibility of creating and providing audiovisual content has grown widely, which has led to an enormous variety of topics in which, some of them, could be considered inappropriate for certain audiences. This is the reason why there have just been done works related to the field of video content analysis and detection of violence.

CPM: no entiendo esta notación, ¿está bien transcrita?

CPM: ¿es necesaria la th?. Yo diría where x_i is the *ith* vector in the feature space

CPM: aquí falta o sobra una llave

In some cases, audio and image features have been combined to address these problem [41]. However, it has been found that sound information could be really useful and a more efficient way of working compared to image, since it is easier to process and the cost is lower. Related works have utilized audio features in the time-domain and in the frequency-domain, similar to the ones explained for ASC, then combined with a normal SVM classifier [15]. Other researches have tried more complicated models with the intention of improving the classification task. It is the case of using DNN, fed with both image and audio data, which performs the task more efficiently [42]. Violence detection has also been used for other applications such as video surveillance. For example, one of the scenarios for this purpose consists on preventing violent acts inside elevators [43]. For this case, the considered dangerous situations are composed of anti-social actions that are likely to happen in this kind of places, concretely, urinating, vandalism and attacks on vulnerable victims, such as women, children or elderly. The framework proposed is based on audio-visual data, but the master classifier is driven by audio, due to the possible subtleness of the scenes that are desired to detect. So, first the audio incident detector triggers the process when a non-silent event takes place. Then, the image processing begins in order to extract information related to who is involved in the action and how aggressive is it. Another utilization of the surveillance approach is its use for the evolution of smart cities [16]. For this goal, since the system will be implemented in real-life environments, one of the advantages about working with data coming from sounds is the respect for privacy, that, otherwise, using video recordings would be violated.

The difference in these two applications, apart from the task they are addressing, resides on the data they are working with. For violent content analysis, the data usually comes from fictional audio sources as movies or video-games. However, for real-environment systems, the data is extracted straight from actual day-to-day life situations. In this second case, some disadvantages can be appreciated. For example, the signals are not preprocessed, which means the original properties of the sound are not modified so the processing part before classification becomes tougher. Also, the presence of background noise is more common and loudness of some events, as speech, may vary with time [44].

2.3.1. Gender-based violence

Throughout history, women have been subjected to abuse and suffering in many different situations even though in those that were considered as their close familiar environments. They have been bashed, sexually harmed and psychologically misstreated by those who were supposed to be one of their closest intimates [45].

In the same way, in the recent times, late studies have shown that 35% of women from all over the world have been victims of physical or sexual damage [46], and 43% of women from Europe have declared going through some psychological or mental violence at least once in their lives [47]. In this context, it is necessary to define the concept of gender-based violence, which can be described as the set of harmful behaviours that are

focused on women and girls just because of their sex, such as female children and wife abuse, sexual assault, dowry-related murder and marital rape, among others. Particularly, violence against women involves any act of verbal or physical force, extortion or lethal denial which has a woman or girl as a target and provokes the physical or psychological hurt, humiliation or irrational privation of liberty and contributes to continue women subordination [48]. Within this definition, it can be considered that most of the times that these violent situations take place, they are originated due to persons that are supposed to be part of the victims' closest circle of trust, i.e., their husbands or boyfriends. This is called Intimate Partner Violence (IPV) and it is recognized as a public health problem affecting women across their life span resulting in different undesirable unhealthy outcomes, such as depression, chronic pain and even dead [49].

2.3.2. Our point of view

As a contribution to the EMPATIA-TC[1] project developed by Universidad Carlos III de Madrid, the main goal in this work is to make progress in detecting gender-based violence situations, specifically applied to day-to-day scenes, in which IPV is likely to be present. One of the parts from the proposed system is composed by wearable devices that the victim can carry to collect diverse types of information and process them to obtain conclusions and increase the efficiency. Among these accessories, we can find a pendant that senses the user's voice and the surrounding audio to analyse what is happening at a given moment. For our purpose, the interesting part resides on achieving auditory data so as to detect violent incidents that consists of sounds already known for characterizing these episodes considered dangerous by the victim.

The definition that is assigned to violence is really important in order to define which audio events should be taken into account. However, considering the subjectiveness of this concept, categorizing violence for every type of user is an extremely difficult task. For this reason, the final idea to answer this question is to make the victim able to decide which kind of hearing events the system must be aware of. In the complete project, this can be carried out by a phone user interface which displays a list of sound events and she has the labour of picking up those that are violent according to her criteria. Since the development of this tool is out of the scope of this work, we have decided to implement a simpler mechanism which will be explained in subsection 4.2.1.

2.4. Databases

A main objective was to find a database that allows for building a system with the desired characteristics, so a rich variety of acoustic events is needed with an essential big representation of violent sounds. In the table 2.2 is represented a relation of the different

[1]**title** funded by Department of Research and Innovation of Madrid Regional Authority, in the EMPATIA-CM research project(reference Y2018/TCS-5046)

databases that have been considered for the realization of this work.

The last three options shown in table 2.2 are the ones that better suit the problem of this project. *VSD Benchmark* was the first option we chose. Within the two ways of working given, the movies and the YouTube videos, the former was the easiest to use since the annotation specified exactly what kind of violent events were present in the scene and the onset and offset time stamps within the whole film. However, this data is copyright restricted and it was necessary to pay for the contents. The latter was accessible but just indicated the presence of violence, without determining the type of event. Another choice was the *Freesound dataset* because it contains all types of videos so we could extract those classes that are more interesting for us. However, it is still under an annotation process and it is not ready to download yet. As a final conclusion, we decided to go for *AudioSet*, which **will be explained further on. CPM: recuerda poner la sección específica.**.

Name	Description	Considerations
URBAN-SED [50]	10,000 soundscapes with sound events. Every soundscape contains 1 to 9 sound events with strong annotations.	Events are completely specified but it just contains three interesting types of classes.
UPC-TALP [51]	It belongs to CHIL project, for the AED task. Isolated acoustic events that occur in a meeting room environment.	Payment is needed to achieve the data and the classes are a little out of our topic.
MIVIA: Audio Events Data Set for Surveillance Applications [52]	6,000 events with background noise.	The classes included belong to our topic, but they are just three: glass breaking, gun shots and screams.
TUT rare sound events [53]	Source files for creating mixtures of rare sound events (classes baby cry, gun shot, glass break) with background audio.	Similar problem to MIVIA: just from three interesting classes.
IEEE AASP Challenge [54]	Composed by ASC and AED. It is formed by two subtasks: OL (Office-live) and OS (Office Synthetic)	Labels for both subtasks are out of our scope since they are likely to happen in an office environment: keyboard clicks, hitting table, etc.
TUT-SED Synthetic 2016 [55]	Isolated sound event samples were selected from commercial sound effects	The variety of classes is large enough but for our purpose just four of them are useful.
VSD benchmark [56]	Violent events from 32 Hollywood movies and 86 YouTube web videos, together with high-level audio and video concepts.	Payment is needed to purchase the movies and the videos do not specify the type of violent event
AudioSet [57]	An ontology of 632 audio event classes and a collection of 2,084,320 human labeled 10-seconds sound clips from YouTube videos.	Our final choice. Plenty of the videos have more than one audio label but we were able to adapt the data to the problem because of the huge amount of clips.
Freesound dataset (FSD) [58]	Filling AudioSet ontology with 297,144 audio samples from Freesound.	This may seem a very good option as well but it is not available as of today.

Table 2.2. TABLE OF STUDIED DATABASES

3. RESOURCES AND METHODS FOR ACOUSTIC VIOLENT DETECTION

In this chapter we are going to explain the different resources we have chosen to create an approach that allowed us to implement our final models. First, we include a detailed description of the database that we have chosen in section 3.1. Then, we comment and analyze the system we are going to employ so as to obtain high level features for our classification experiments in section 3.2. In this point, the model is itemized step by step and also a theoretical explanation is included in 3.2.1.

3.1. Database: AudioSet

3.1.1. What is AudioSet

Audio Set can be described as a large-scale sound dataset with the goal of putting the availability of audio and image data on the same level. It is composed by a huge variety of 2,084,320 manually-annotated audio events in video format and it is organized by following an ontology formed by 632 different audio classes. The data has been extracted from YouTube videos and the labelling process has been based on diverse factors such as metadata, context and content analysis. It has been developed by Google with the purpose of producing an audio event recognizer that can be applied to plenty of acoustic situations coming from the real world [57].

How cite? CPM: como misc, por ejemplo en la página web y con el artículo que la describe

Due to some limitations found during the building process that will be explained along the section, the current final release of the dataset is composed by 1,789,621 segments, where each video usually last 10 seconds which results in a total duration of 4,971 hours of video and audio. After executing the selection process in which the different labels were populated with the final corresponding segments, a total of 527 classes were gathered, out of which 485 counted with at least 100 samples.

3.1.2. Ontology

In order to put this dataset together the events have been organized in an abstract hierarchy. This is composed by higher-level classes which describe a certain type of sound event and also act as parents of other labels that refer to more specific events. With this purpose, the relationship among different classes needs to be non-exclusive, so labelling similar audio events may result into a more general class, the parent, if there was ambiguity. This is also helpful for labellers due to group the clips in an easier and faster way.

The Audio Set Ontology was collected considering some fundamental guidelines explained below:

- A complete collection of all desirable labels was prepared so that it can be used to define sound events from real-world aural data.
- When labelling an audio event the result must match the criteria of a common listener.
- Different categories should be easy to distinguish by an ordinary listener. In the case that two different labels do not satisfy this requirement, these should be merged. With this condition, the plethora of possible labels remains limited.
- The distinction of two different classes must be done by relying just on the audio, it cannot be accompanied by image or visual information.
- The hierarchy should not be very deep by keeping the number of children per parent class to no more than 10. This also eases the annotation labour.

It is easy that an ontology of this volume gets leaned or biased towards a particular direction due to several factors, such as the subjectiveness of its creators or the selection of the initial set of classes used when starting to label. With the intention of generating a primary list that covers a wide range of audio events in an objective way, the researchers decided to apply an impartial, web-scale text analysis from the very beginning. They agreed on detecting hyponyms of the word "sound" by utilizing a modified version of the famous technique called *Hearst patterns* [59], a method proposed to automatically acquire hyponymy lexical relations from unrestricted text. As a result, an enormous collection of terms came up. This was filtered by considering how well these terms represented audio, i.e. by combining together the global frequency of occurrence and how exclusively these are recognized as hyponyms of "sound" instead of other terms. As an output of the final process, a list of 3000 terms was obtained.

With respect to the hierarchical relation among categories, it was constructed by the authors with the main intention that this satisfied their human comprehension of the sounds. Event though this is a subjective manner, it also makes sense since this is how the hierarchy best performs its labour on helping human labelling. After all the organization process, the model is not based on a strict organization as a singular node can appear in many different locations, i.e., a single node can be child of different parent nodes. As we said before, the final result is composed by 632 audio event labels and, in the hierarchy, the deepest class is six level away from the top parent. Figure 3.1 shows the nodes that belong to the two top layers of the ontology.

This whole structure is offered to any user as a file in JSON format. A couple of fields have been included for each label in order to describe its meaning and make clear its position within the hierarchy. A description for all of these can be found in table 3.1.

For the field *ID*, the identifiers are known as Machine ID (MID) and belong to the *Knowledge Graph* designed by Google [60]. This is a knowledge base that Google services use to improve the quality of its search results and it is composed with information

Fig. 3.1. First two layers of Audio Set Ontology [57]

ID	This field includes the Knowledge Graph Machine ID (MID) that best describes the sound or its source. It is used as a primary identifier for the class.
Display name	Short name formed by one or two words that identifies the audio class. It sometimes includes a small alternative separated by a comma so it does not feel ambiguous.
Description	One or two explaining sentences so the meaning of the category is more defined. These can be extracted from Wikipedia or WordNet.
Examples	At least one example of the label is provided as a URL of a YouTube video.
Children	An array filled by the Machine ID (MID)s from all immediate children of the class.
Restrictions	It specifies if the category in question either has been discarded or there are no audio clips under it.

Table 3.1. FIELDS PER CATEGORY IN THE ONTOLOGY.

extracted from a wide variety of sources. The MIDs are the identifiers of the different elements that belong to this huge dictionary. For instance, the MID of the word "Speech"

is "/m/09x0r". Another field that deserves a special explanation is the one corresponding to *Restrictions*. Within all the categories of sound events, there are two flags that indicate an exclusive behaviour that differ from a typical label: "blacklisted" and "abstract". The former refers to a class that has been hidden from labellers due to its confusing meaning. The latter has been used for those classes that are just employed as intermediate nodes in order to provide a better grouping inside the organization, and are not expected to be used in the implementation tasks. In total, out of the 632 categories, 56 have been categorized as "blacklisted" and 22, as "abstract".

3.1.3. Data access

The data can be obtained through the website [61] in two different formats:

- Files in .csv format that include for each video segment its YouTube video ID, start time, end time and the one or more labels it belongs to.
- Instead of the audio files themselves, they provide already extracted audio features for each segment in compressed files that are available in TensorFlow (TF) [62] record files that can be easily downloaded.

Together with these two options of obtaining the data, a neural network model is also provided in the database website which can be used to extract embeddings as high-level features. It is called *VGGish* [63] and it has been pre trained on a preliminary version of the database YouTube-8M [64]. A more detailed explanation of this system is later included in 3.2.3.

Based on the first choice, we did some attempts of downloading a couple of videos from YouTube using the information given in the .csv files. Then, we decided to try to extract our own embeddings features by running the VGGish network with the downloaded videos in order to see how they look and check their behaviour on a classification task. Also, for the second option, we downloaded the provided embeddings already extracted by the Google developers and see if they were also useful for our problem. A little experiment explained in 4.2.3 were implemented so as to check if the two manners of obtaining the embeddings were equivalent.

3.2. Feature extractor

VGGish model design was based in an already existed network called VGG. In order to describe the embedding features extractor we used, it is first needed to make a clear explanation of what a Artificial Neural Network is and how Convolutional Neural Network behave. Then, an explanation about how the VGG model is implemented included. Then, we can move to analyze the differences and changes included in VGGish.

3.2.1. Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN)

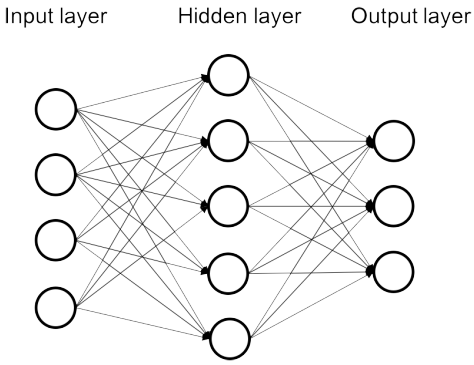
Before going deep in the description of CNN, the core idea of ANN must be explained. As it was already mentioned at the end of subsection 2.2, these are based on the communication way that the neurons follow inside a brain to interpret the stimuli they receive. This is actually the hidden concept behind the ANN intention, which is transforming input data into a desired output so that it can be used to perform a given task.

A basic architecture is usually composed by an *input layer* of neurons, a certain number of *hidden layers* and an *output layer*. The model receives some input, typically a feature vector, and pass it trough the hidden layers in order to perform some transformations. The neurons of a layer are completely connected to the neurons of the next layer but neurons in the same layer are not usually connected. For this case, we can say that they are Fully-Connected (FC) layers. Finally, the output layer gives the resulting outcome of the network [65]. This is just the basic and initial nature of the idea but it was implemented in many different ways depending on the input data and the desired task.

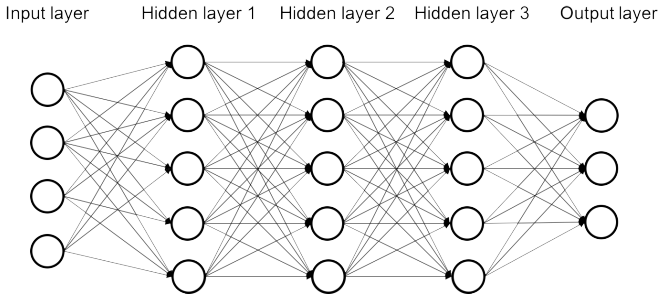
Within the different possibilities of designing an artificial neural network, there are two main options. The selection of one of them depends on the complexity of the problem, the format of the input data and how it is combined with other techniques. One of them covers all the models that are structured with a shallow architecture. An example of this would be a basic type of network known as multi-layer perceptron (MLP) that follows the structure explained for ANN with Fully-Connected layers but composed just by one hidden layer. This type of systems have performed well in many tasks but they have shown some limitations in more complicated applications [66].

When working in real-world cases such those that involve natural signals, such as human voice, natural sounds and visual scenes, for example, it is needed the development of a deeper architecture in order to extract and understand the complexity inside this type of data so as to build. For example, visual and speech recognition tasks are exploited with system that follows the hierarchical transformations that the humans perform when understanding this type of data. reliable and useful internal representations. This type of models are collected inside the Deep Neural Network (DNN) field. DNN has been one of the most studied fields since it was proposed in 2006 [66]. In figure 3.2, it can be found a comparison between shallow and deep models.

One of the most explored implementations inside the DNN is the one called Convolutional Neural Network (CNN). This maintain the basic idea of receiving some inputs and apply a dot product operation to them before subjecting the data to a non-linear function but with the difference that input data are expected to be images [68]. The architecture of a Convolutional Network is based on the organization of the Visual Cortex in the human brain. This means that single neurons responds to some stimuli only in a certain region of the visual field, the Receptive Field. Then, a collection of these fields overlap with each other so as to cover the full visual space [69].



(a) Shallow architecture



(b) Deep architecture

Fig. 3.2. Comparison between shallow networks and deep neural networks [67]

About the input data, it is introduced as a matrix instead of as a row vector. Then, the architecture of the layers adapt to the fact that input data are images. They do not consist of a single row of neurons, but they have a shape constituted by three dimensions: heigh, width and depth [68] . In this way, the net can capture the spatial and temporal relations present in an image by applying some filter functions. With this construction, the model can be trained with a smaller number of parameters and also reuse the weights.

include image?

A Convolutional Network can be described as a sequence of layers in which every layer converts a volume of activation results from the previous layer to another by computing a differentiable function. The architecture of a CNN is always composed by three type of distinguishable layers which are *Convolutional*, *Pooling* and *Fully-Connected*. A normal architecture will follow the next succession of layers [68]:

- INPUT: this represents the image at the entrance of the whole model. It commonly is a 3D matrix , one dimension per channel of colour, and contains the values of the original pixels. In the case of a RGB image the number of channels should be three, for example.
- CONVOLUTIONAL: considering the neurons that are connected to a region of the input image, this layer computes their output by performing a dot product between the weights and the value of the pixels. The depth of the resulting data will coincide with the number of filters used.

- POOL: this performs a fixed operation that consists of a downsampling of the spatial dimension. It typically reduces by half the height and the width of the input image.
- FC: this is the final layer which gives the output of the whole net. It is a vector-shape layer and is completely connected to the previous layer. In a classification task the number of neurons matches the amount of classes of the problem.

Ask about normalization

Also, another type of layer can be optionally added in order to improve the performance of the model. This is called the normalization layer. It is a good practice in some cases in order to put a limit for the layers output. When using an unbounded activation function, in order to bound the result of the output layers, the normalization process can be used before the activation operation is applied. Two of the most common techniques are Local Response Normalisation (LRN) and batch normalization [70]. However, in many cases the contribution of these layers has been almost null for the performance of the model [68].

Note that this list represents the order of the layers in the architecture, however, the number of layers of each type vary depending on how the model is designed and how deep it is desired to be. Some examples of deep architectures can be found later in section 3.2.2, in table 3.2.

With this building, a Convolutional Network passes the information of the pixels layer by layer while reducing the size of the input images in a way that is much easier to process. This is performed without missing any feature since all of them are essential to compute a good prediction outcome.

In the convolutional layer, a convolution operation is performed for each small region of the input image. The element that carries out this operation is called *kernel* or *filter*. This partial region that the kernel covers in the input data is set by deciding the height and the width as a designing parameter. About the depth, it must match the depth of the input data, for an image this can be understood as the number of channels, so the filter can slide trough the image along its other two dimensions. The, the results are obtained by computing the convolution between the pixels inside the region and the values of the filter. An example can be observed in figure 3.3. For each filter, a 2D activation map is obtained after subjected the result of the convolution to a non-linear operation. One of the most common functions used for activation is the Rectified Linear Unit (ReLU) function. Finally, we will pack all these in a unique volume that will consist in the output of the convolutional layer [68]. For example, if we have a number of 64 filters, we will get then a group of 64 2D activation maps that will be the input of the next layer. Traditionally, it was the first convolutional layer the one that was responsible of capturing low-level features that can be related to edges or colours in the image, for example. The upper layers were the ones in charge of extracting high-level features so the network was able to understand the image similar to how the humans do [69].

We have included how the input interacts with the convolutional layer and also intro-

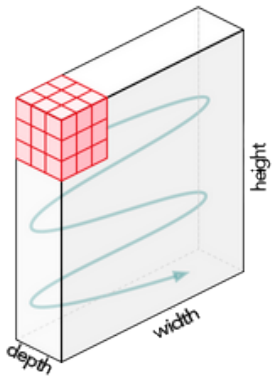


Fig. 3.3. Movement of the kernel represented as a red block along the width and the height of the original input image. It is clear that it finally occupies the whole depth of the input image. The arrow indicates an approximation of the movement that the filter follows [69].

duced the concept of depth of the resulting volume. However, it is needed to explain how the total size of this output is computed and what it depends on. There are three main parameters that play a role in this decision: *depth*, *stride* and the *zero-padding* [68].

- DEPTH: it is a hyperparameter that corresponds to the amount of kernels used in the convolutional layer when learning from the input. The set of neurons that extent through the depth dimension operating in the same region is called depth column. This is the previous mentioned red block shown in figure 3.3.
- STRIDE: this indicates the displacement of the sliding filter along the image. If it has a value of 1 (non-strided), this means that the kernel will shift a position of one pixel along the width. When it is set to 2, then it moves two pixels. It rarely has values greater than 2. The moving process consists on the filter going from right to left with a hop determined by the stride value. When the right margin is reached, then it jumps to the beginning of the next row from the left margin and repeats the process in the same way until the image is completed [69]. The bigger the stride, the smaller the size of the output volume.
- ZERO-PADDING: this is also called just *padding*. It consists on adding zeros to the borders of the image. What is a hyperparameter in design is the size of this padding. This allows to control the width and height of the output volume so we can keep it the same through the different layers. When it is indicated a "valid padding", it means that no padding is added and the output size will be reduced. If we indicate "same padding", then it has a value of 1 and means that the resulting volume will have the size of the input [71].

As shown below, the total size of the obtained volume, D, can be computed as function of the input size, W, the spatial 2D dimensions of the convolutional filter, F, the stride of the filters shifting, S, and the amount of padding in the zero-padding, P [68].

$$D = \frac{W - F + 2P}{S} + 1$$

The convolutional layers can be grouped all together, one after another, but they are usually interpolated by what it has been previously defined as *pooling* layer. This habit has the objective of reducing the width and the height of the resulting volumes in a progressive manner in order to decrease the total number of training parameters and control the computational cost, and, therefore to avoid overfitting [68] . There are two types of pooling operation: the max-pooling and the average-pooling. The former just returns the maximum value from the portion of the image where the filter is placed. However, the latter computes the average of all the values in this portion [69]. The pooling layer acts in an independent way on every input depth level.

Include definition of overfitting or just cite it

Two hyperparameters are needed in order to configure the spatial portion in which the pooling is computed: the filter size, related to length of height and width, and the stride. Also, for this operation zero parameters are introduced since it consists on a fixed operation over the input data. In figure 3.4, it is shown how the max-pooling operation is performed [68].

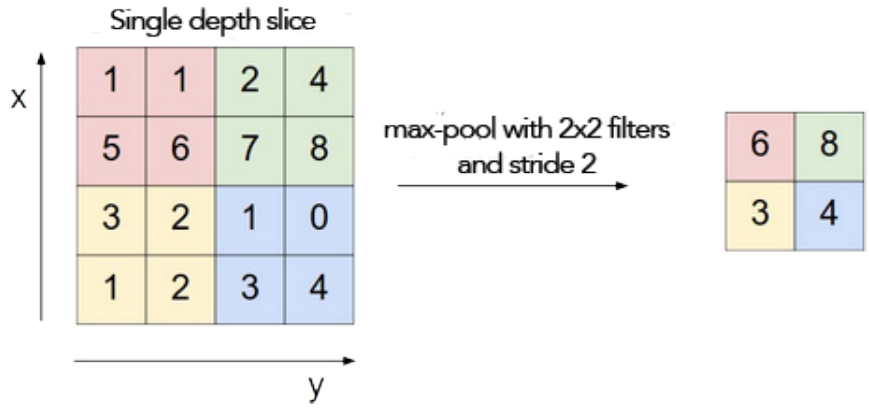


Fig. 3.4. Max-pooling operation with a filter size of 2 and a stride of 2 for a single depth filter. Each colour represents the action portion for each operation [68]

Together a convolutional layer and a pooling layer constitute a typical complete layer structure in a Convolutional Network. The number of these complete layers can vary depending on the design and the task needs. These represent the main core of the network before passing to the last layers and calculating the final output [69].

For the final step, a common way of acting is to apply some non-linear activations in order to learn from the high-level features resulted from the output volume of the last convolutional layer. This takes place in the Fully-Connected layers ending the model. This part has the shape of a Fully-Connected layer, as the one described in the basic model of ANN. To make the weights from the convolutional layers available to feed this last part of the model, it is necessary to flatten the output volume and obtain a column vector. The learning process is possible due to a backpropagation operation performed in

every epoch of training. As a total output, the values that represent the classification of the different samples into one or another category are obtained [69]. A common classification technique is the one called *Softmax*. This normalizes the result of the previous FC layer into a vector whose values follow a distribution that total sum results in 1. This type of output is called *soft predictions* and represent the probability for each sample of belonging to a category in the classification [72]. However, there exists other activation functions that can compute the classification output in other ways.

In figure 3.5 an scheme of a common Convolutional Neural Network is shown. This takes an input image, compute the features along three groups of convolutional layers plus max-pooling and, finally, includes three Fully-Connected layers, being the last one a softmax function layer with as many neurons as classes that gives the soft predictions for this image of belonging to each class.

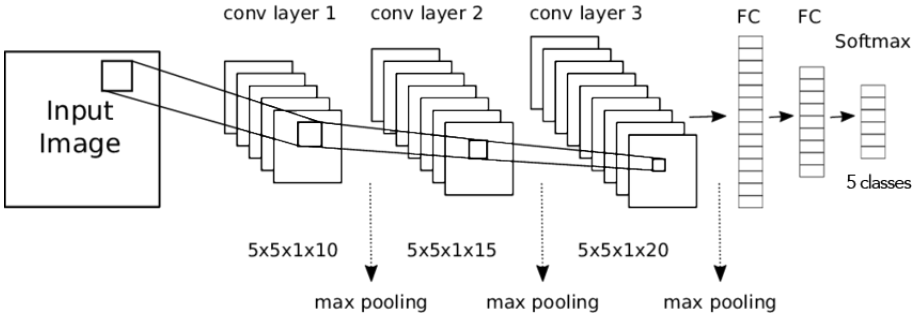


Fig. 3.5. Representation of a typical CNN model [73]

Convolutional Neural Network (CNN) were originally designed to work with images but they have been applied to other fields as audio. An image can be defined as a matrix of values, i.e. pixels, in two or three dimensions, so the only prerequisite to use a CNN is to have the input data in this form. For audio researching, it has been commonly used the log-scaled mel spectrogram, that is a way of representing audio in a scale different than frequency that is more similar to humans perception. It has been used in plenty of works with CNN and also combined with other techniques [50] [74] [75].

3.2.2. Visual Geometry Group (VGG) model

CNN are usually able to achieve really good results and even improve human skills on Computer Vision tasks, for example, on recognizing object in an image. With the exponential growth of the researching works on this field, some challenges have appeared so as to promote the creation of new systems and test their efficiency and results. This is the case of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), based on the database of the same name, ImageNet. As a solution for the proposed exercise, the investigators from the Visual Geometry Group (VGG) in the University of Oxford implemented a new system achieving the first position and winning the challenge in 2014 [76]. The work they proposed consists of a study of the depth in a Convolutional Network architecture and how this can affect to the accuracy on the goal of large-scale image

recognition [77]. To try this, it was necessary to increase the number of layers in the network, which was viable due to use a small size of convolutional filters in all of them.

For the training step of their system, they used an input image with standard size of 224×224 in RGB format. The principles to build the architecture are detailed below:

- The input image crosses a bunch of convolutional layers in which the kernel has a size of 3×3.
- The convolutional stride has a value of 1 pixel.
- The padding is fixed to 1 pixel, so the dimensions of the input do not change during the convolution.
- Max-pooling is also included with a window size of 2×2 and a stride value of 2 pixels.
- Two Fully-Connected (FC) layers with 4096 channels after all the Convolutional Network layers.
- One FC layer with 1000 channels to perform the ILSVRC classification.
- A soft-max layer is employed for the final layer.
- All hidden Convolutional Network layers use the non-linear function ReLU as activation function.

All the designs that the creators came up with are based on these initial guidelines, except from just one case where Local Response Normalisation (LRN) is applied, previously mentioned in subsection 3.2.1. They just differ from each other on the number of layers, starting with 11 the first approach and ending with 19 the last one. The different architectures are specified in the table 3.2 [77] and are ordered from A to E.

3.2.3. VGGish model

The model we used in our project for feature extraction presents a configuration with principles similar to the ones explained in the previous subsection 3.2.2 but with slight changes that their developers have included in order to adapt it to the audio approach. As previously mentioned in subsection 3.1.3, the implementation of this model has been obtained from the website of the database Audio Set, in which a code repository is public available [78] and all the information about the implementation is included.

The architecture is based in the configuration A from table 3.2 with 11 weights. The differences respect to the original network are listed below:

- The input size was changed from 224×224 to 96×64 because of the log-mel spectrogram audio inputs.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 3.2. VGG CONVNET CONFIGURATIONS.

- They built the implementation with just four groups of convolutional and max-pool layers, instead of five.
- For the last FC layer, they decided to build it just with 128 channels since it is the layer that compacts the final embedding and defines its dimensions.
- Also the final *Softmax* layer is not used.

Figure 3.6 shows how the configuration of the final VGGish model looks.

Input stage

Before passing the data through the whole CNN, a preprocessing stage has been included by the developers in which the input audio will suffer some transformations.

1. In the first place, after loading the input audio, the sample rate and the number of

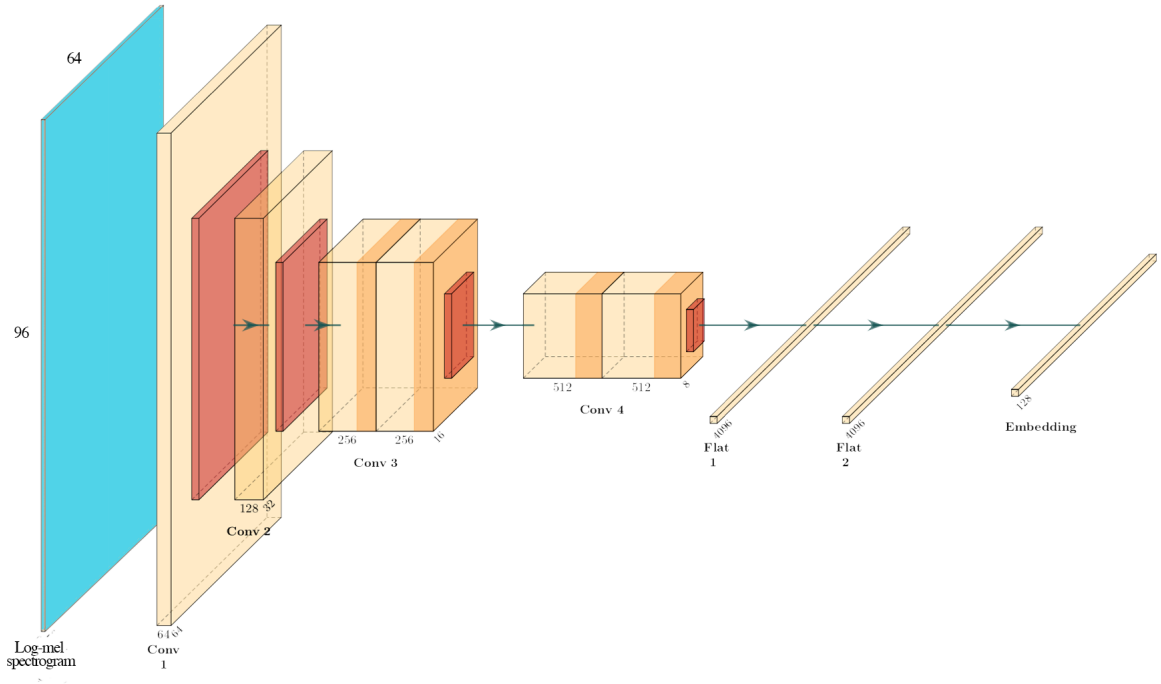


Fig. 3.6. VGGish architecture

channels are checked to be 16 kHz and monochannel, otherwise the file is transformed to satisfy these conditions.

2. When the data is ready, the next step is computing the spectrogram. For this, it is necessary to calculate first the Short-Time Fourier Transform (STFT) of the signal. The operation is performed by using a sliding Hanning window of 25 ms with a hop size of 10 ms. As a result, just the magnitudes that correspond to the positive frequencies are retained, since the ones corresponding to the negative part of the spectrogram are redundant.

3. Transforming the spectrogram to Mel scale is what follows. To do so, they compute all mel frequency bins that are going to modify the values of the spectrogram in frequency domain by following a hertz to mel conversion formula, that can be found in appendix **??**. The result is a mel spectrogram from 125 Hz to 7,500 Hz divided in 64 bins.

4. Then, the log-mel spectrogram is calculated by doing the logarithm of the previous result plus a small offset value to avoid the logarithm of 0.

5. As a final step, they compute a framing operation over the log-mel spectrogram. The resulting are non-overlapping examples of 0.96 seconds, in which 64 mel bands and 96 frames are contained, each frame with a duration of 10 ms.

Therefore, the result obtained is an ensemble of 10 frames, approximately one per second, each of them with size 96×64, i.e, 96 frames and 64 mel bands.

Embedding stage

Once the initial processing part is done and the log-mel spectrogram matrix is computed and divided into the desired number of frames, it is used as input data for the VGGish CNN. After all the computations inside the network, each example is converted into an embedding of size 128 giving a result of one of these per second of the original audio file.

Postprocessing stage

As final step, they performed some post-processing of the resulting embeddings. A Principal Components Analysis (PCA) [79] transformation is done joint with a whitening process. Also, a quantization to 8 bit for each embedding element. All these actions are computed with the purpose of making the final output compatible with the embeddings obtained from the YouTube-8M database.

Transfer learning and why choosing this type

4. OUR APPROACH FOR ACOUSTIC VIOLENT DETECTION

For our final approach when implementing this system of acoustic gender-based violence detection we first need to select the classes we are going to use for the classification so as to obtain the corresponding embedding features with VGGish from the audio data, obtained from the database explained in section 3.1. Then, a supervised multiclass classification is performed for the different events and, finally, the soft output from this step is used as an input for a binary classifier so we can differentiate the audio from violent and non-violent.

In this chapter we want to make clear why we think that the VGGish network is a good option for our problem in 4.1. Then, the procedure when starting to prepare the data and to extract the feature embeddings with this model is included in 4.2. Also, the demonstration of the similarity between both types of embeddings previously mentioned in 3.1.3 is described in 4.2.3. Finally, the last section 4.3 includes an explanation of the different algorithms that we use for the experiments explained in chapter 5.

4.1. Why VGGish

For our goal, one of the toughest tasks consisted on the selection of data that properly adapted to our problem and its preparation so as to obtain features that allowed us to characterize every acoustic event from a violent point of view. Since our first efforts of finding an available dataset containing enough violent scenes was driving us to a dead end, we decided to take advantage of a huge database which let us rethink the standpoint about how we were going to address the problem. As it was mentioned in subsection 2.3.2, one of the main questions was how to define the term violence for each victim depending on her certain situation. After finding the Audio Set database, previously explained in section 3.1, with all its variety of samples, we had a wide range of audio data to work with. This is how we came up with the system explained below in 4.2.1.

At this point, not only we had an idea but we had already found a data resource to start with. However, the issue was related to what kind of features could be extracted in order to categorize events from different nature with a unique violent label. An acceptable conception of the term violence could be expected to cover all kinds of impulsive events such as hits, smashes, gunshots, yells, etc. Also, we would like to introduce sounds that were likely to happen in a domestic environment within a tense atmosphere as children crying, dog barking or glass breaking. We also wanted to take into account the possibility of including other cases not usually considered violent a priori. For example, the sound of keys jangling or the noise produces by a shaver machine. These situations may be too particular and just would be present in few uses, but this is how we understand the problem given the advice of the social sciences experts in the UC3M4Safety team.

So, our first intention was to apply some audio processing techniques to extract low-level features, as the ones previously explained in **??**. Even though there are plenty of previous works and a lot of tools to work in this way, it was not sure which path should we take in order to decide what features better fitted our task. Apart from this, since the database had such a big volume it would have supposed an enormous cost of time to compute features every time we wanted to try new type of categories. Moving on, by following the advances on finding new level features already mentioned in **??**, we decided to investigate new methods of extraction based on the use of Neural Network models. Nevertheless, even thought the features obtained in this case had been more appropriated, the time consumption of training a big model was one of the aspects that did not totally convince us.

The previous selection of Audio Set as our dataset allowed us to get to know the VGGish model proposed by Google researchers for feature extraction. This system loads the parameters already learnt from training with another huge dataset as YouTube-8M. This is possible due to apply transfer learning idea, explained above in subsection 2.2.1, that consists of leveraging features or weights extracted from certain models and use them in simpler ways for different tasks [37], so all the computational cost and training time is not a problem any longer. Finally, we decided to put in practice this pre trained embedding extractor by loading the given parameters so as to obtain our final input representations.

4.2. Our approach

In order to start describing our approach, we will first explain in 4.2.1 how we obtained the input data to work with by using the resources previously explained in 3.1 and 3.2. Then, we will move to the implementation of the whole model in chapter 5.

4.2.1. Input data

Different phases took place when trying to obtain all the necessary data from the YouTube videos specified in the database. We will explain them from the first step of deciding which classes better fit our problem to the last part in which the desired embeddings are achieved.

Violent classes

In subsection 2.3.2 we mentioned our idea about giving the victim a choice of defining her own perception of violence, so the final implementation can adapt to her situation in a better way. To do so, we have taken advantage of the ontology provided by the Audio Set creators we explained in subsection 3.1.2.

Our system has been implemented based on the idea of using the parent-children rela-

tionship among the different nodes to ease the process of selecting among the 632 classes. It must go through all the branches so as to offer the victim the possibility of choosing any of the audio event categories. However, instead of consider each label individually, this starts the way from the parent classes down to the children ones.

Let's say we begin from the class "Human sounds" that is the top level of all sounds emitted by humans contained in the dataset. The system will ask the user if she wants to advance in that direction, i.e., to go across the branches that belong to that part of the tree. If the answer is positive, it would go for the next class, that in this case it would be "Human Speech". It will advance this way until there were no children nodes in the actual class. When this happens, the user will be asked to add the label to the record of *violent classes*, that will be saved in a text file so they can be read by other parts of the model further on. If the user does not want to go deeper, she will be asked to add the current class to the record. If the answer is "No", then they system will jump to the next sibling category. The corresponding flowchart is shown in figure 4.1.

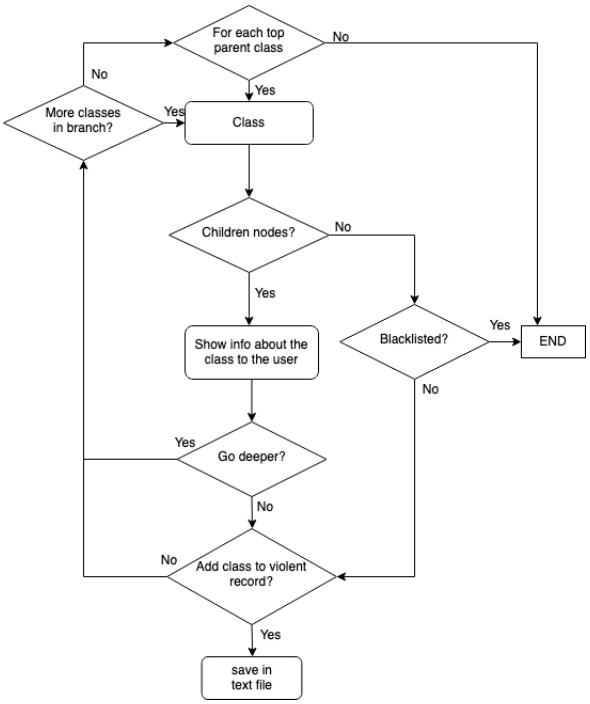


Fig. 4.1. Flowchart about selecting violent classes

This way of flowing through the different classes allows to skip those parts that are not related to our problem. For example, as we can see in figure 3.1, one of the top parent labels is "Natural sounds", that relates to sounds from weather phenomena. In most of the cases, these classes will not be selected so the whole branch can be skipped.

Downloading videos

The following step consists in achieving audio files that belong to the chosen labels. For this purpose, we have made use of the .csv files that were explained in subsection 3.1.3. For each included video, we took its ID and build the corresponding YouTube URL. Once downloaded, we trimmed the file considering the onset and offset and, finally, converted it to audio format (.wav). In our script, we can pass as a parameter the identifier of the desired classes in comma-separated format and either the number of videos per class for a balanced set or a total number of downloads for an unbalanced set. However, there might be some errors when obtaining all the data: the two most common cases are due to lack of enough videos of the desired type in the dataset or because the video is not available anymore on YouTube. When this happens, a message is shown to the user.

It it also worth mentioning that throughout the developing of this downloading task, a script has been coded to achieve the whole dataset in both formats, video and audio, for future works. We are not going to specify anything else since this feature was not finally used.

4.2.2. Extracting embeddings

At this point, all the desired data has been already downloaded to extract the embedded features that will be used to train the model. For this part, we have used the VGGish network explained in subsection 3.2.3. Since the audio files duration is usually 10 s, and the embedding extractor gives as a result a vector of size 1×128 for each second, we will obtain a 10×128 feature matrix composed by values within the range 0 - 255. Therefore, our input feature matrix will have a size of $(number\ of\ audios) \times 10 \times 128$. The corresponding labels will be stored in a vector of size $(number\ of\ samples)$.

There are some points about the data obtained in this step that should be commented. One of them is about the length of the audio files. As it was previously indicated in 3.1.1, most of them last 10 s because the creators decided to set this duration for the audio events, but this can change if the video is originally shorter. For these situations, since the model is configured to have an input of 10×128, the embedding matrix of the shorter clip must be padded with zero-rows to achieve the required dimensions. Even though this does not happen very often, there might be some silent segments that will be labelled with the category of the rest of the audio. A solution for this problem explained later in 5.1 was implemented.

The other case is related to what was explained in 3.1.3. For the recent explanation of how to extract the input features from the audio files we have utilized the first manner of accessing the data, i.e., by reading the .csv files with the videos information before downloading. There is a second option of using the already extracted embedding features. However, these do not look exactly the same when comparing them to the ones obtained by running the VGGish with the downloaded videos. This difference is because the implementation of the given code differs from their internal production system in computing issues such as underlying libraries in the installation of VGGish and hardware equipment. In spite of this, the result in classification tasks are expected to be equivalent. In order to prove this assumption, we decided to try a small system with both kinds of data. The reason for this check is that although, for this work, the development of the methodologies could be made by using the first and faster manner of accessing the data, for the real implementation in the pendant device within the EMPATIA project (out of our scope) only the second way will be feasible.

4.2.3. Assessing the differences between the two types of data access

In order to check what is explained above in 4.2.2, we have decided to run a little experiment in which a toy classification is performed. Also, we wanted to visualize the different features to check if we could appreciate common patterns by using the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm, which will be explained later in this section.

Our first step consisted in determining a subset extracted from the original dataset. We thought about choosing for this small application a set composed by three classes that could be considered violent and other three that were non-violent. Apart from this, we paid attention to the number of samples per category to pick some classes over others. Finally, we ended up picking up the labels detailed in table 4.1 and a number of 80 samples for each of them, which led us to a total of 480.

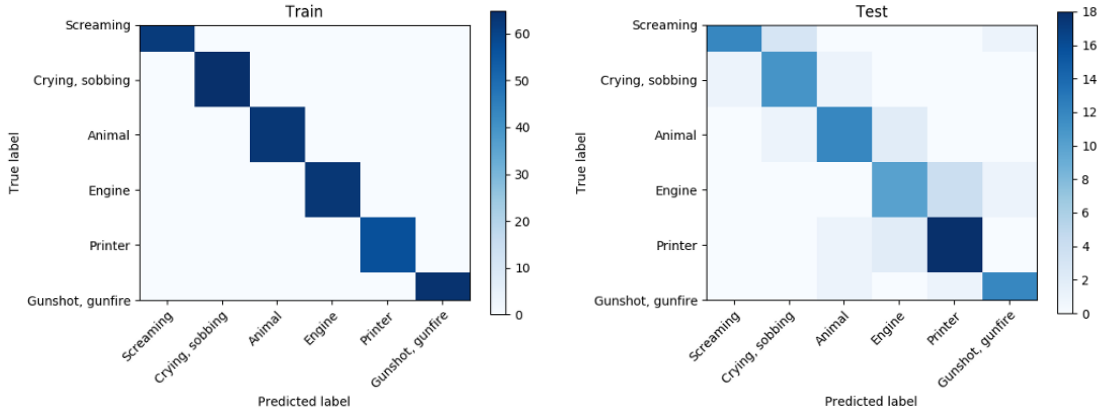
Class	**Description**
Screaming	A sharp, high-pitched human vocalization; often an instinctive action indicating fear, pain, surprise, joy, anger, etc. Verbal content is absent or overwhelmed, unlike Shout and Yell.
Crying, sobbing	Sound associated with the shedding of tears in response to an emotional state, arising from slow but erratic inhalation, occasional instances of breath holding and muscular tremor.
Gunshot, gunfire	The sound of the discharge of a firearm, or multiple such discharges.
Animal	All sound produced by the bodies and actions of non-human animals.
Engine	The sound of a machine designed to produce mechanical energy. Combustion engines burn a fuel to create heat, which then creates a force. Electric motors convert electrical energy into mechanical motion. Other classes of engines include pneumatic motors and clockwork motors.
Printer	Sounds of a computer peripheral which makes a persistent human readable representation of graphics or text on paper or similar physical media.

Table 4.1. CHOSEN CLASSES FOR A SMALL CLASSIFICATION. *SCREAMING*, *CRYING, SOBBING* AND *GUNSHOT, GUNFIRE* ARE CONSIDERED AS THE VIOLENT ONES.

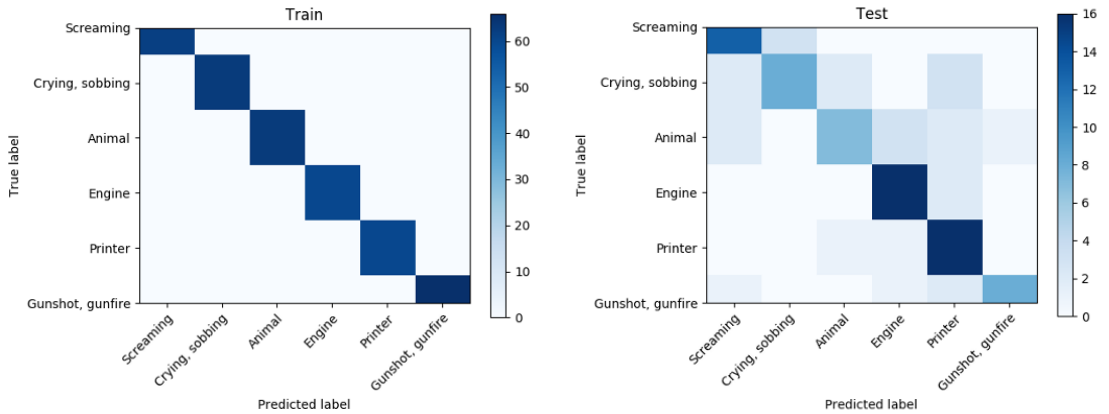
For the classification task, we decided to create a small CNN composed by few layers. Since our input data are matrices of shape 10×128, these were treated as images so the

model was built with layers that perform spatial convolution [80]. The architecture is detailed in figure 4.3. We used a small kernel size of 3×3, zero padding so as not to change the shape of the output and a activation of function ReLU. Two dense[2] layers are added at the end, first one with also ReLU as activation function and the second one with *softmax* to perform the classification, and as many filters as the number of classes.

In order to measure the results, since our subset is balanced, we could have evaluated our model by computing the accuracy and the confusion matrix. More information about these metrics can be found in the appendix .1. In figure 4.2, the four confusion matrices corresponding to the training and test phases for both types of data are shown. Also, in table 4.2, it is included the accuracy for each case. The results are more accurate when training with the embeddings extracted directly from the audio files we downloaded. When obtaining them from the TensorFlow files, they value of the metrics indicate a worse performance. However, we opted for this second manner because the performance difference is not too big and it requires much less computational cost and time loss.



(a) Confusion matrices when embeddings are extracted from audio files



(b) Confusion matrices when embeddings are taken from *.tfrecord* files

Fig. 4.2. Confusion matrices

[2]Dense is a also a common form to refer to Fully-Connected layers. Since each neuron is fed with an input from all the neurons in the previous layer we can say that they are densely connected [81]

format / subset	**Train**	**Test**
Audio file	0.98	0.63
.tfrecord	1.0	0.72

Table 4.2. ACCURACY VALUES FOR AUDIO AND *.TFRECORD* FILES

For the training phases, it can be appreciated that there may be an overfitting since the accuracy is perfect. This means that the NN stop improving its capacity of learning how to solve the problem in a certain moment of the training task. Instead, it does learn some behaviour pattern that the training data follows. This impacts negatively in the model since the new data that the system will have to learn from will look different and will not follow these same rules [82]. In spite of this result, we did not give it so much importance since we just wanted to do a quick check of the similarity between the two types of data which can be appreciated due to the similarity of both metrics results.

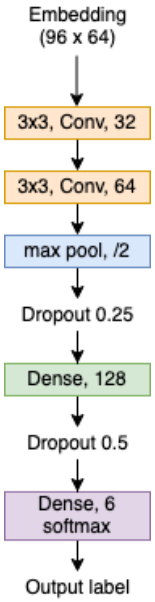


Fig. 4.3. Architecture to see how the different embeddings work

Dimensionality reduction for visualization

Apart from the classification exercise, we wanted to see if by plotting the data samples we were able to identify or appreciate some common patterns. In our problem, each of our samples is characterized by a matrix of features with 128 columns, which means that we were working with data belonging to a high-dimensional space. Visualizing this type of data has always been a case of study for many different fields. Plenty of methods have been published so as to find a solution for this task. Some of the most accepted consist on reducing the dimensionality of the data so this can be transformed from the high-dimensional space to a lower one and can be visualize in a common scatter plot of 2D or 3D. These techniques rely on the idea that within a multivariate sample denoted as $x_i = [x_{i1}, ..., x_{in}]^T$ and considered to be a point that corresponds to a $n - dimensionality$ space, a $d - dimensionality$ space can be found, so that $d < n$. If this is possible, then the observations can be transformed to this lower dimensional space d without any loses [83].

One of the most common and well-known dimensionality reduction methods is the one known as Principal Components Analysis (PCA). This follows the idea previously explained. It specifically aims at extracting the *important* information from the original data and transform them into a set formed by orthogonal variables which are actually

known as principal components. This is done by multiplying the matrix data X by a projection matrix Q that contains the coefficients of the linear combinations that allow the conversion. The projections must be orthogonal to each other and they represent the data maximum variance in descending order, being the first component the one with largest variance [79]. In fact, each of the projections correspond to an eigenvector in descending order following the value of the eigenvalue. So, the first component will be the eigenvector with the highest eigenvalue. It has been proved to be one of the most appropriated options in dimensionality reduction nowadays. Its use is completely accepted and it is implemented in many famous software libraries. However, it presents some limitations. One of them consists on just considering linear combinations of the original data. When the relation is non-linear, a dimensionality reduction with this technique may result in a loss of information [84].

For these cases, other methods have been developed such as t-distributed Stochastic Neighbor Embedding (t-SNE). This algorithm appears as an extension of the previously published Stochastic Neighbor Embedding (SNE) [85]. Both are based on the same idea of a new way of measuring the similarity between samples. Instead of comparing two observations, let's call them x_i and y_j, by computing the euclidean between them, this is done by calculating the conditional probability $p_{j|i}$ of x_j being picked as a neighbour of x_i considering that the samples belong to a Gaussian distribution centered at x_i. This depends on how far the samples are from each other, i.e., it is high when they are close and minimum when there are totally separated [86]. Apart from this, two analogous observations are created in the subspace of lower-dimensionality, y_i and y_j, and conditional probability $q_{j|i}$ is computed in this situation. It is important that, for y_i and y_j to be faithful representations of x_i and x_j, both conditional probabilities must be equal.

In order to calculate the probabilities, a crucial factor is the variance of the Gaussian distribution. There is not a unique valid choice for this parameter, so t-SNE performs a binary search so as to find the optimal one [84]. This is also influenced by what is called the perplexity. This can be defined as an assumption of the number of adjacent neighbours for each point. It is a value that is fixed by the user, but it usually belongs to a range from 5 to 50 [86].

There are several considerations that should be known before looking at a representation of data from this algorithm [87]. Actually, it is not an easy task to understand this kind of plots, since the distance between points in the new subspace are not related to the real euclidean distance, which has been denoted as "The Crowding Problem" [86]. This means that the groups cannot be interpreted as real collections of data in the original dimension. However, in order not to misunderstand the data distribution, a couple of visualizations varying the parameters usually tends to be done so that the conclusions can be based on more than one result. In figures 4.4 and 4.5, we show ten t-SNE outputs, five for each type of data for diverse values of perplexity from 2 to 50, respectively.

Fig. 4.4. t-SNE results for wav format with a legend that shows the labels of the corresponding samples in the original 128D space

The selection of the perplexity value depends on the number of observations per class [87]. Since our subset has 80 samples for each category, we can consider that a proper value is 20 or 30. As we can see, a boundary cannot be extracted among the different labels, but we can see some grouping patterns in the data that correspond to the original labelling and helps us to confirm the similarity of the two different given types. It is true that this method should never be used as an algorithm for clustering itself, but it is a good resource as a backing strategy to other results, as in this case.

CPM: aumentar las figuras para que se puedan leer las leyendas.

Fig. 4.5. t-SNE results for *.tfrecord* format with a legend that shows the labels of the corresponding samples in the original 128D space

4.3. Methodology

Debería poner ejemplos de aplicación al audio como hice en las CNNs

4.3.1. Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE consists of an over-sampling method whose function is to generate synthetic samples in order to populate the minority class in a classification problem by making use of the proper observations that are already contained in the data [88].

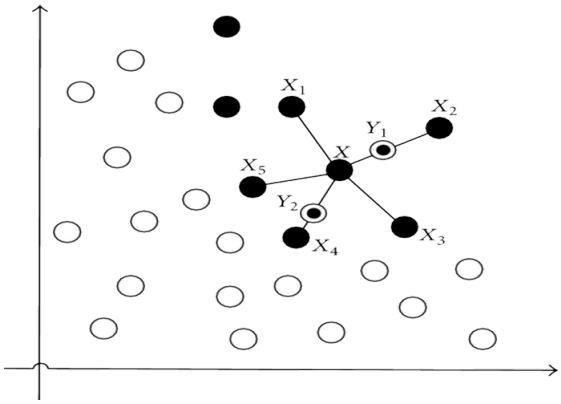


Fig. 4.6. SMOTE algorithm [91]

Essentially, this technique works by choosing samples from the less populated class that are represented in the features space, placing a line between two of these samples and then select a point inside the line at a random position. First, an arbitrary sample is picked and then a number of k closest neighbours in the feature space are selected from the same class. One of the neighbours are randomly chosen and a synthetic sample is generated [89]. The number of neighbours k can vary depending on the amount of over-sampling needed. The standard value which is the one proposed in the publication paper is $k = 5$ [88].

For the generation of the synthetic samples, the distance between the feature vector first arbitrarily selected and one of the nearest neighbours is computed. Then, the resulting value is scaled by multiplying it by a random number that lies in the uniform distribution $U(0, 1)$. Finally, this new sample, S, is added to the initial feature vector and placed in between these two observations in the feature space [88]. S can be defined as follows:

$$S = x + u \cdot (x^R - x)$$

where x is the arbitrary selected sample the first time, u is the random value from $U(0, 1)$ and x^R is one of the closest neighbours [90]. In figure 4.6, it is shown a graphical representation of the algorithm. In this case, X is the original sample for which the closest neighbours are represented as X_i where $i = 1, 2, 3, 4, 5$. The generated samples are Y_1 and Y_2 that are placed in the line segments.

The common way of applying this technique is in the train set before performing the fit of the model. Usually, an under sampling is performed in the most populated classes and, then, the data augmentation is done for the minority ones [89]. Our implementation follows this way of acting and it is explained in section 5.1.

4.3.2. Support Vector Machine (SVM)

Considering instances of data with their respective ground-truth labels, this classifier finds the frontier defined as a hyperplane that maximizes the distance between the two observations closest to each other that belong to different classes and are more likely to be

misclassified. These two points are called support vectors [28]. In order to find this hyperplane, the data is mapped into a higher dimensional space so it can be more separable. This is done by applying a kernel function that eases the process of finding a proper mapping function. Some of the most used ones are the polynomial, Gaussian, linear, sigmoid and RBF [92].

This type of method is characterized for being a binary classifier. However, its implementation has been transported to multiclass problems by combining various SVMs in order to establish a decision criterion so the whole system can chose among Q categories. This can be done by following either one-versus-all or one-versus-one approach. The former addresses the problem training Q classifiers so as to differentiate between data from one class and data from the other $Q-1$ classes. The latter treats the problem by training $\frac{Q(Q-1)}{2}$ SVMs to distinguish among all possible combinations of categories. In any of the approaches, the classification of a certain observation is performed by computing the distance between the sample data and the hyperplane that defines the frontier [1].

The one-versus-one is the one implemented by C-Support Vector Classification (SVC) from Scikit-learn library and is the one we will use in one of the experiments explained in section 5.2.1. In figure 4.7 it is shown a representation of the decision boundaries that could be designed by other type of methods against the hyperplane a SVM classifier would draw.

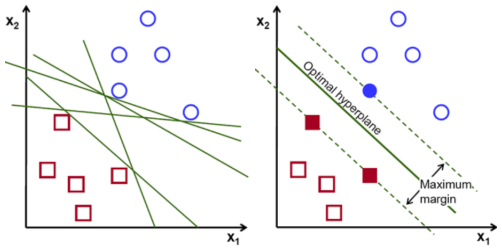


Fig. 4.7. Visualization of a hyperplane set by SVM and other decision boundaries [93]

4.3.3. Recurrent Neural Network and Long Short Term Memory

Human learning does not happen at each moment independently in the sense that every time something new is learned, it depends on the previous knowledge in order to interpret it. This can also be explained by saying that our understanding is persistence.

Traditionally, Artificial Neural Network are not able to act in this way. They **cannot use the time** as a property to infer conclusions or predictions from the previous instances that belong to a sequence of data. To address this task Recurrent Neural Network (RNN) have been developed, so they can take information and make it persistent. In figure 4.8, it is shown a block of a RNN in which the network, A, is fed with an input x_t that is actually a sequence of data. The output is the value h_t. An inner loop takes the information from the outcome and pass to the input for the next learning step. This can be easily seen in the unrolled part of the right, how the information flows from one element to the next one

[94].

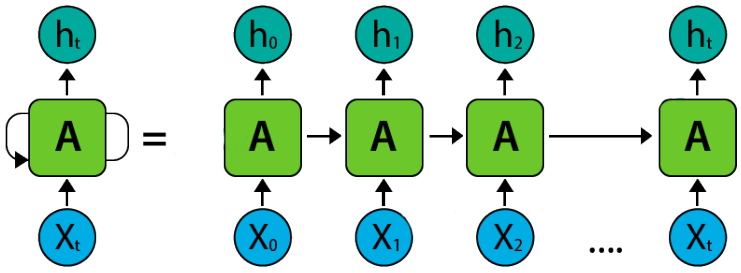


Fig. 4.8. Scheme or a recurrent neural network about how the loop works

This type of networks has became the standard when dealing with sequential data. They have been applied in many tasks, such as language modelling, speech recognition, translation, etc. However, the most simple approach present a problem when the information that must be considering when long-term dependencies. For example, if the task consists of predicting a new instance based on the previous ones in a not really long sequence in which the dependency resides on close instances, normal RNN can be used and work in the way explained above. When the distance inside the sequence between the predicted element and the ones with the important information that this new one depends on is too large, the network cannot learn this connection [94].

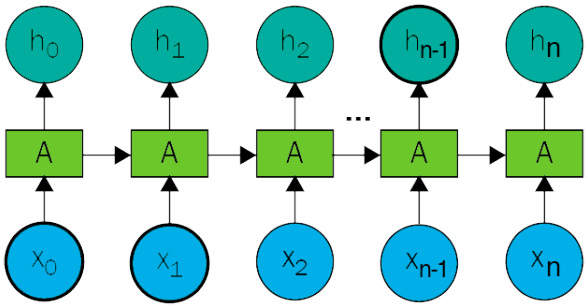


Fig. 4.9. Example of long-term dependencies situation. If the predicted output h_{n-1} depends on x_0 and x_1, the vanishing gradient problem may appear.

This happens due to the value of the gradients becomes very small during the back-propagation in the learning process. Once the loss function for the new predicted output is calculated, this must be propagated through the rest of the network in order to update the weights. In a typical neural network model, **the neurons that get updated are those from the hidden layer directly previous to the output one**, but, since the data is a sequence, the neurons from all the earlier layers must be updated as well. The actual problem appears when renovating the value of the weights through time, i.e. updating the weights used to connect the hidden layers in the unrolled temporal loop. So, if the value of the gradient becomes very small the updating process start to be null when finding the new values of the weights and the model stop learning. This is known as the vanishing gradient problem [95].

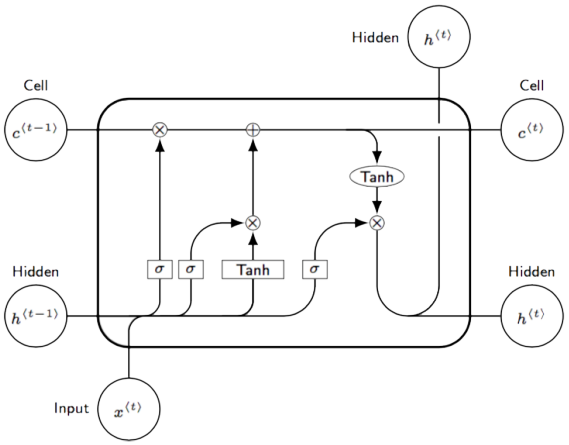


Fig. 4.10. LSTM module with the representation of the different operations that take place inside of it

As a solution, a new type of RNN was developed and named as Long Short Term Memory (LSTM) networks. These can be defined as a recurrent network that can learn long-term dependencies, so the vanishing gradient problem is not an issue for these models. Nowadays, they are widely used in several fields, since they can remember information along long periods [94]. The main structure along is the same shown above in figure 4.8. The difference from the way of working in a common RNN is the inner architecture in the neural network. In LSTM, the *A* modules are defined as shown in figure 4.10. The notation used is as follows: the rectangles denotes a neural network layer, the circle shapes refer to point-wise operations and the arrows means vector transferring.

The main concept of a LSTM network resides on its cell state and the different gates. The cell state, represented in the diagram by the upper arrow, is the one in charge of transferring information all the way through the chain of modules. It can be thought as the memory of the whole network. Its objective is to just carry information considered relevant for the model. This way is how the LSTM brings the information from the first layers to the last ones without suffering the vanishing gradient problem. The content of the cell state is modified by the point-wise operations that acts by following the gates criteria. These are in fact neural networks that has the goal of deciding which information can be treated as relevant and so added to cell state. This process can be understood as a learning/forgetting stage [96].

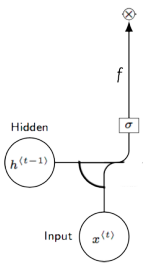


Fig. 4.11. Forget gate

In order to understanding how this process works, we are going to address the interaction of the different gates in the learning stage.

The first step is performed by what is called the *forget gate* layer. It must decide which information must be included in the cell state that travels through the time chain. This is way the layer contains a sigmoid function that outputs values between 0 and 1, in which 0 means dropping the information in the previous cell

state and 1 implies to keep it. As shown in figure 4.11, this is done by paying attention to the hidden state which is the output of the previous module, $h^{<t-1>}$, and the input of the current one, $x^{<t>}$ [96]. The resulting function f is defined as follows, where W_f are the weights of the RNN and b_f the bias.

$$f = \sigma(W_f[h^{t-1}, x^t] + b_f)$$

Next, the decision about what information from the current input $x^{<t>}$ must be taken to the cell state is performed. The gate that computes this operation receives the name of *input gate*. This part can be divided into two different steps. First, the previous hidden state,$x^{<t-1>}$,and the current input feed a RNN with a sigmoid for the activation function, which takes the decision of **which values must be updated** by converting them into a range limited by 0 and 1. This result denotes the importance of the value, being 0 non-important and 1, important. Then, also $h^{<t-1>}$ and $x^{<t>}$ are used as input for the tanh layer so the values are mapped between -1 and 1. This last activation function is included in order to **avoid the vanishing gradient problem previously mentioned, since a function whose second derivative takes more time to tend to zero is needed**. Finally, both outputs are multiplied before the updating process in the cell state [96]. The equations for the output of each RNN module are included below, where W and b denotes the weights and bias for each layer [94].

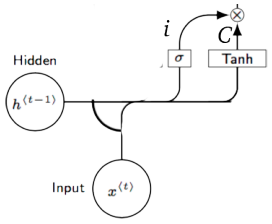


Fig. 4.12. Input gate

$$i = \sigma(W_i[h^{t-1}, x^t] + b_i)$$

$$C = \tanh(W_C[h^{t-1}, x^t] + b_C)$$

In the step 3, the output of the multiplication from the input gate layer and the output f of the sigmoid function from the forget gate, modify the old cell state from the previous module, $c^{<t-1>}$, so it can be updated. For the outcome of the step 1, a point-wise multiplication is performed, $f * C^{<t-1>}$. Then, to the output of this operation is added the resulting product of the input gate, $i * C$. This two operations can be translated as a forgetting and a learning stage. First, the values the forget gate decided to remove from the cell state are actually forgotten. Then, it must learn the new values belonging to the current input also weighted by their importance, what denotes how much the state values are going to be updated [94]. The new cell state expression c^t is included below and the process is shown in figure 4.13.

$$c^{<t>} = f * C^{<t-1>} + i * C$$

Finally, the last step of the module corresponds to the *output gate*. This has the function of deciding how the next hidden state is going to be like. First, $x^{<t>}$ and $h^{<t-1>}$ are used as input for a layer with a sigmoid activation function and the output o is obtained.

Also, the cell state $c^{<t>}$ feeds a point-wise tanh so its values are clipped between -1 and 1. Both outputs are multiplied. The o result decides what values form the current input must be kept in the future hidden state,$h^{<t>}$. The actual output of the module is the hidden state. It also transports the relevant information updated together with the cell state to the next network. The outputs o and h^t can be defined as shown below. Also, in figure 4.15, the stage of this last part is included.

$$o = \sigma(W_o[h^{<t-1>,x^t}] + b_o)$$

$$h^{<t>} = o * \tanh(C)$$

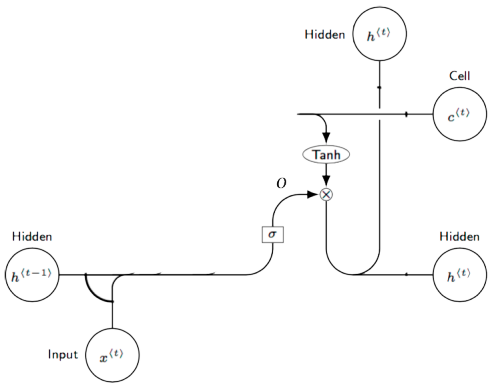


Fig. 4.15. Output gate

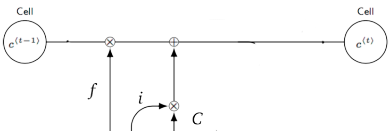


Fig. 4.13. Cell state

5. EXPERIMENTS

In this chapter we are going to explain the final models we chose for our work and the input data preparation to feed these models with. For the different solutions, we have used the algorithms previously explained in the methodology section 4.3. Below, a list including the 6 implementations compared follows:

- SVM classifier for multiclass classification
- SVM multiclass + SVM for a final binary classification
- LSTM for multiclass classification
- LSTM multiclass + SVM for a final binary classification
- CNN for multiclass classification
- CNN multiclass + SVM for a final binary classification

5.1. Input data preparation

For the proposed experiments, we have decided to build a small dataset with the embeddings extracted form the *.tfrecord* files that belong to 14 classes: half of them *violent* and the other half, *non-violent*. To find these, we first did a run on the simple user-interaction program that is explained in 4.2.1 as if we were gender-based violence victims. As we said a total of 28 classes were selected. From this set, we picked 7 as the violent classes. The other 7 were chosen just by looking at the ontology to provide negative classes.

It is important to mention that this collection of data is composed by samples with just one label assigned. As explained before in 3.1, the Audio Set database is highly unbalanced. Some of the most appropriated classes to be considered violence are scarcely populated. When doing the selection of categories, apart from paying attention to the own meaning of the class, we also checked the number of samples. For the non-violent type, this was not a problem, since we took some of the most populated labels. However, in the violent case, we had to deal with the requirement of representing violence and also having enough observations. So, due to these limitations, we could not obtain a naturally balanced set. In table 5.1, the 14 selected labels are shown. In figure 5.1, a bar plot shows the number of samples obtained per class right after the selection.

Some initial preprocessing steps were performed before passing the data to the different models. As mentioned in 4.2.2, we had to deal with the zero-filling problem, which means that some of the rows of the embeddings matrices are completely zero because the

Violent	Non-violent
Baby cry, infant cry	Printer
Slap, smack	Music
Screaming	Speech
Machine gun	Vehicle
Breaking	Animal
Slam	Dishes, pots, and pans
Yell	Wind

Table 5.1. RELATION OF *VIOLENT* AND *NON-VIOLENT* SELECTED CLASSES

duration of the original video is less than 10s. As a solution, we decided to substitute the zero numbers in all the data with the machine epsilon[3] value.

However, in the two models that use SVM for the multiclass classification a different solution was proposed. As it was explained in subsection **??**, this algorithm needs the input data matrix to be in the form [$number\ of\ samples\ \times\ number\ of\ features$], which differs from the originally shape of our data, [$number\ of\ samples \times number\ of\ seconds \times number\ of\ features$]. For this reason, we decided to reshape our data to the required form which resulted in a matrix of shape [($number\ of\ samples\ \times\ number\ of\ seconds$) $\times$ $number\ of\ features$]. With this conversion, instead of working full audio instances, the data samples became the seconds of those instances. In order to remove zero data, we first checked the amount of zero-rows in every class. Since it was not a very significant portion of the data, we took them out of the dataset.

Once we had our data with all non-zero values, we needed to convert the imbalanced set to balanced one. First, for the classification task, we divided our data into train, validation and test subsets, using a 20% for last one. Then, we decided to exclude half of the samples from the most populated classes from the train and validation sets [4]. In figure 5.2, subfigure (a) shows the distribution of data per class in the dataset for the models that use SVM multiclass classifier. In subfigure (b), the distribution for the other four cases is shown. Then, we wanted to generate new data for those with less observations by applying the data augmentation technique SMOTE, explained in subsection 4.3.1. **This technique deduces a distribution for the given observations and generate synthetic samples within the distribution of each class until even the most populated category. CPM: ¡qué raro! ¿por qué iba a generar de la clase más poblada si esta es la que se toma como tope? No sé...**. Just the train and validation sets were subjected to this conversion process, leaving the test set with the original number of embeddings in their natural a priori distribution. The final shape of the input data for each of the experiments

[3]The machine epsilon value is considered the smallest value that satisfies $1 + \epsilon_{match} > 1$. It is the difference between one and the next closest number that is representable as a machine value [97]

[4]*Speech*, *Music*, *Vehicle* and *Animal*

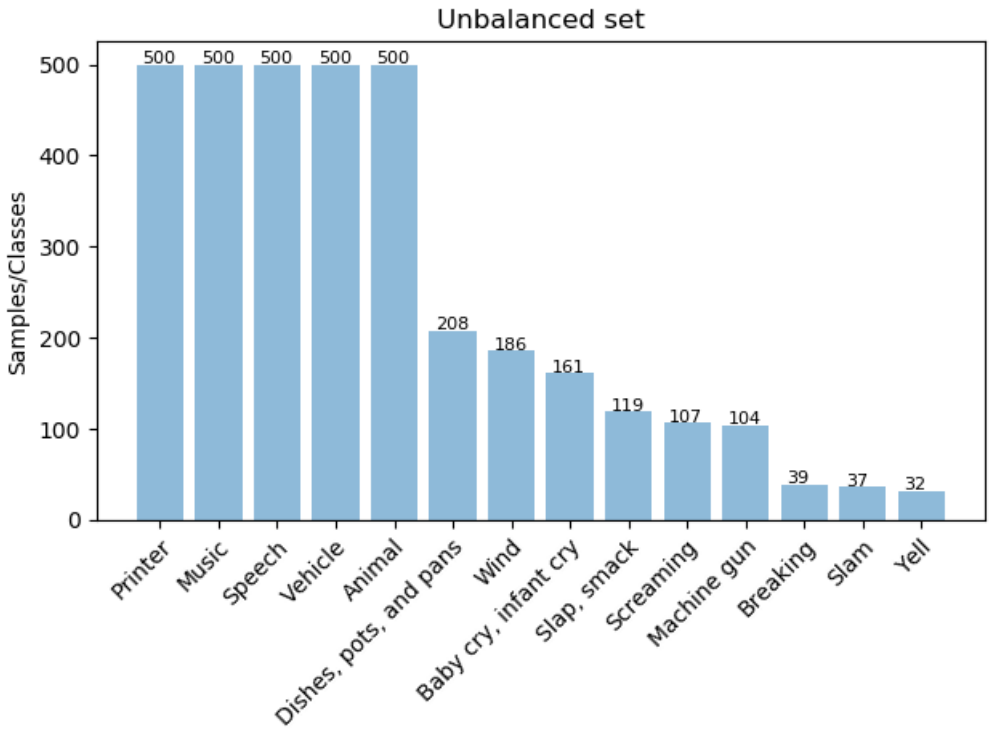


Fig. 5.1. Bar plot that shows the number of samples for each of the selected classes. Clearly, the violent categories are much less populated than the others, that did not have to much any semantic criteria

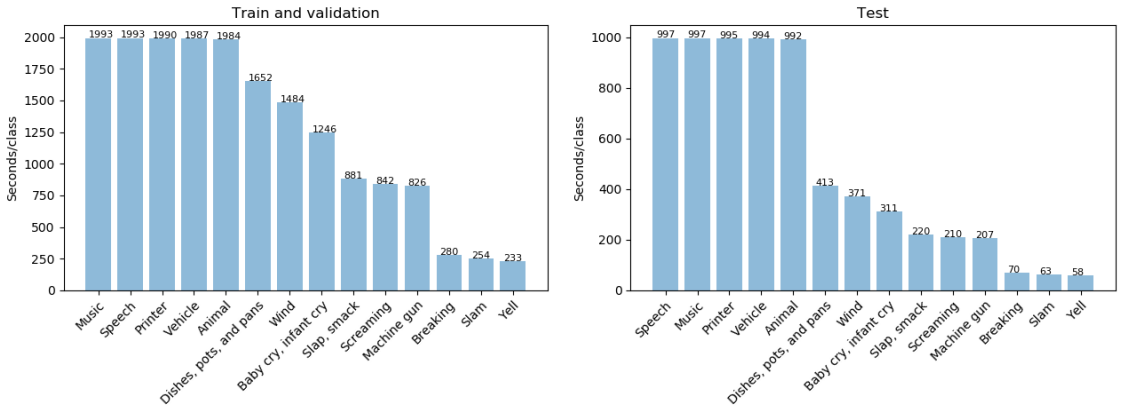
is shown in table 5.2.

	SVM multi and SVM + SVM binary	**LSTM multi and LSTM + SVM binary**	**CNN multi and CNN + SVM binary**
Train and validation	17645 (1993 per class)	2800 (200 per class)	2800 (200 per class)
Test	6898	699	699

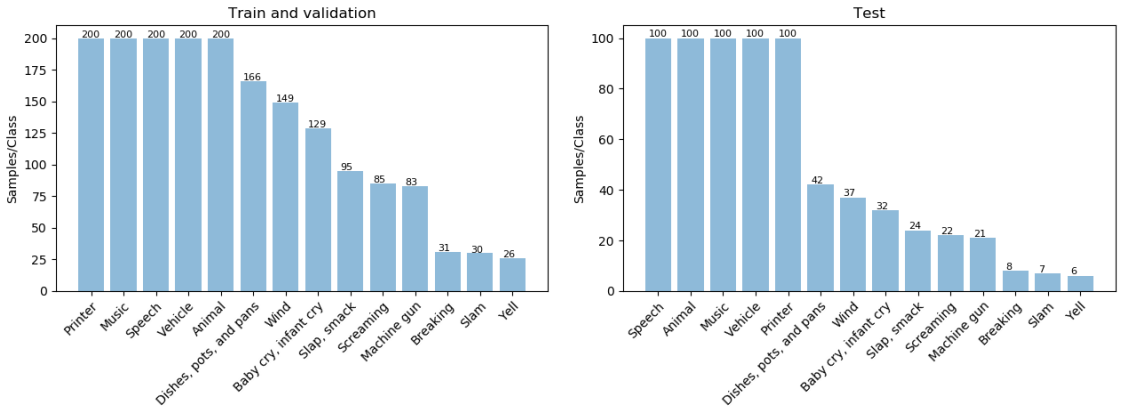
Table 5.2. TRAIN, VALIDATION AND TEST SET FOR DIFFERENT EXPERIMENTS

5.2. Implementations and results

For all the experiments, as mentioned above, the dataset was split by selecting a 20% of the data for the testing procedure. The other part was divided into train and validation subsets with a k-Fold Cross Validation technique, with 10 folds, i.e. 10-fold cross-validation. A more detailed explanation about this resampling technique can be found in appendix .2. The average and standard deviation of the results from the different folds were obtained for train and validation. Then, a final measurement was performed for the test set.



(a) This is the resulting subsets after downsampling the most populated classes and removing the zero-rows for the experiments that involve a SVM classifier



(b) This is the resulting sets after downsampling the train and validation sets. The test set remains the same after the split

Fig. 5.2. Number of observations for train, validation and test subsets used in the different experiments

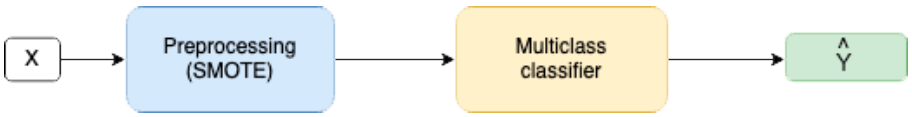
The way of checking the model performance was by finding the accuracy and confusion matrix, whose explanation can be found in appendix .1. For the cross-validation procedure, a matrix with the average values[5] was obtained and also one for the standard deviation[6] so the estimation of the model is represented for the corresponding subsets. Finally, a last evaluation of the model is performed by checking the accuracy just once with the test set. The testing step is performed by using the model for which the highest value of accuracy for the validation fold was obtained during the train process.

As mentioned above, we have in total developed six final implementations. Three of them for a multiclass classification problem that consists of predicting the right label

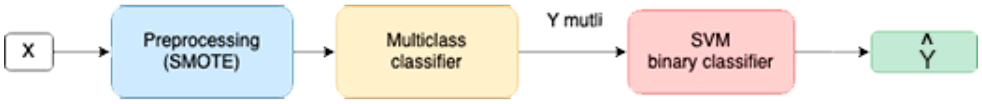
[5]In the average matrices, the results for each cell are shown just with one decimal in order to make a better and more comfortable visualization. However, the colour bar on the right of the plots must be taking into account since there might be some cells in which the value is 0.0 but it is actually greater.

[6]In the standard deviation matrices the results are shown multiplied by 10^2 in order to see the value in the different cells of the matrix

for each sample within the fourteen labels already explained. The other three are basically SVM binary classifiers that take as input the output probabilities of the multiclass classifications. The purpose of these three is to adapt the model to the final objective of creating a model to distinguish between violent and non-violent events. In figure 5.3, two block diagrams are included in order to represent the flow in the multiclass and binary classification approaches.



(a) General model for the the three multiclass classification approaches.



(b) Block diagram that shows the concatenation of multiclass classifier and the SVM binary classifier.

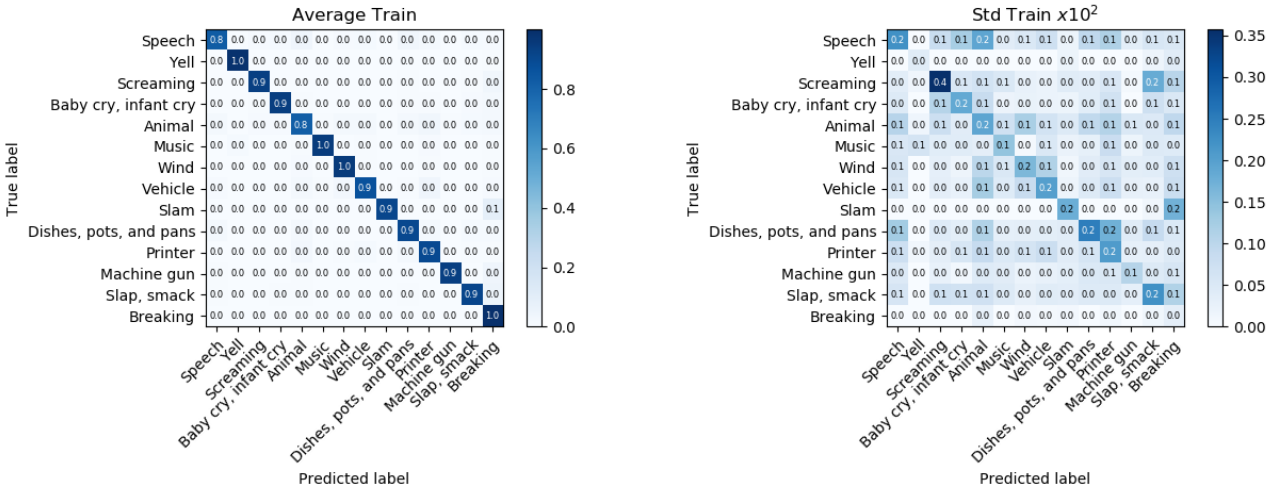
Fig. 5.3. Confusion matrices for SVM multiclass classification for train and validation sets

The output denoted as $\hat{Y}$ *multi* corresponds to the predicted labels by the multiclass classifier. Then, this feeds the binary classifier in order to be trained. The split for this process is also 20% for the test set and the rest is split into train and validation with a k-Fold algorithm. Then, the model instance for which the best accuracy value was obtained in the validation set is used for the testing process.

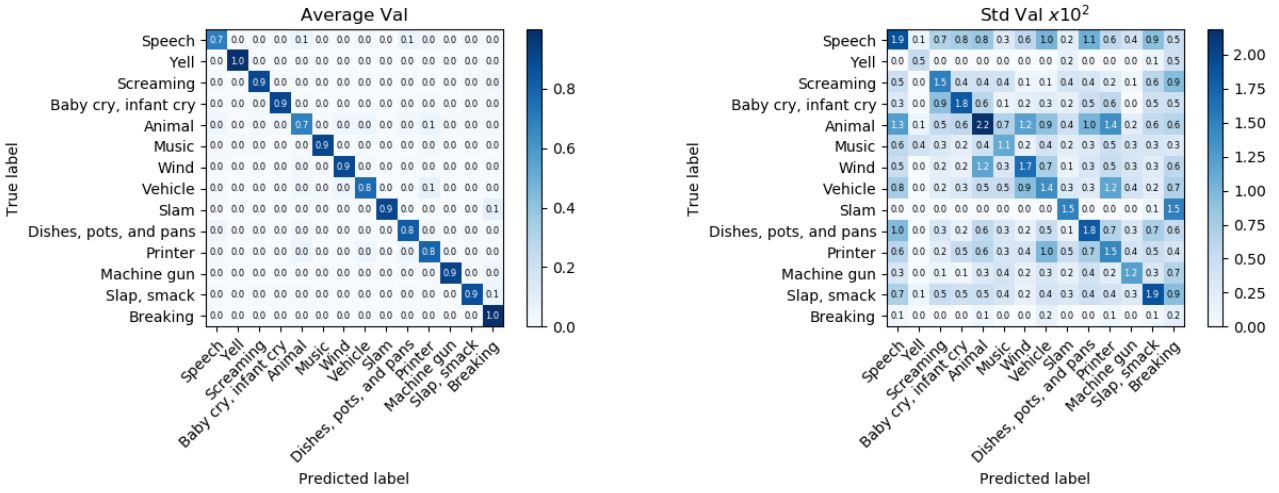
5.2.1. Implementation 1: SVM classifier for a multiclass classification

We decided to establish a benchmark by making a first experiment based on a SVM classifier that is used for a multiclass classification. In this case, we wanted to check the results of the performance of one of the most employed techniques in this filed different from NN. It is true that this method is originally designed for binary problems but, as explained in 4.3.2, we took advantage fo the multiclass algorithm. To define our model, we used mainly the default parameters. In this work, we have not investigated which type of kernel could better fit our problem, however we found in the literature that the default option of RBF is the most appropriated for real world applications [98].

Below, we can find the results for the different sets. In figure 5.4, the confusion matrices for the train and validation sets are shown. In figure 5.5, the confusion matrix for the evaluation on the test set is included. Finally, in the bar plot of figure 5.6, the accuracy values for the three sets are represented. For this case, the standard deviation is not shown for train and validation because it is zero when considering two decimals.



(a) Average and standard deviation for the train set



(b) Average and standard deviation for the validation set

Fig. 5.4. Confusion matrices for SVM multiclass classification for train and validation sets

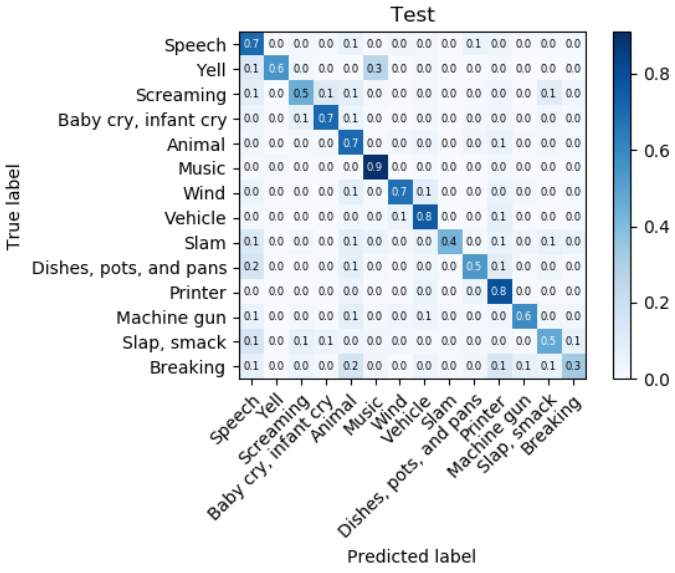


Fig. 5.5. Confusion matrix for the test set for the SVM multiclass classification

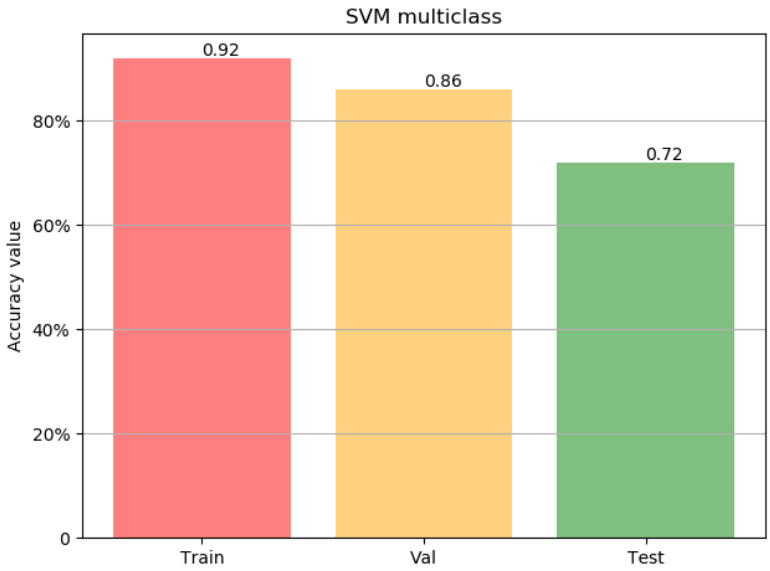


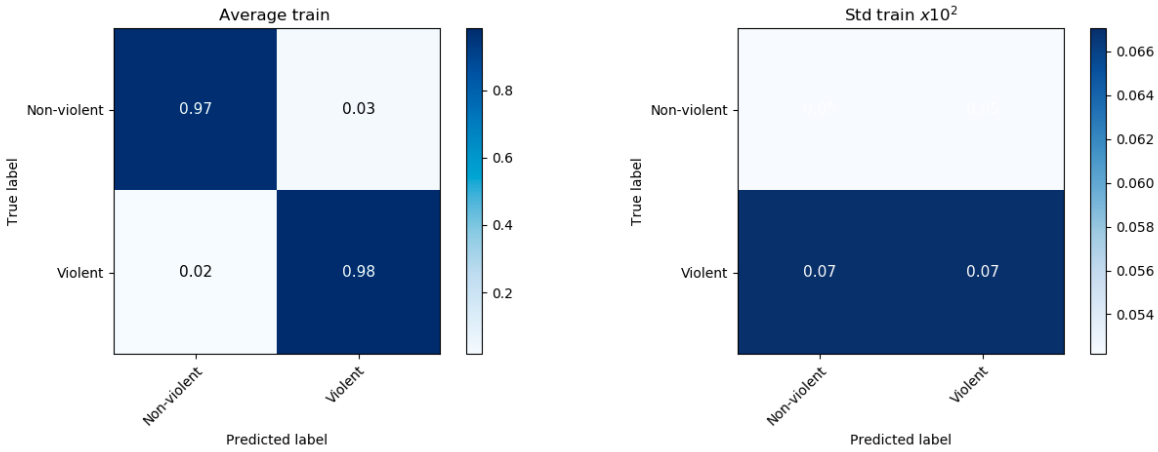
Fig. 5.6. Accuracy values for the three sets for the SVM multiclass classification

As we can see, the values of the average values of accuracy are a 92% for the training set, a 86% for the validation and a 72%. This can also be appreciated in the average confusion matrices. The diagonal for the training and the validation stands out the other cells but they do not present a sense of overfitting due to they are not completely uniform. In the test results, we can see greater values in the true positive sections for the most populated classes, mentioned above in section 5.1. This makes sense since the test set has more samples in these categories. We can consider a good result for this case. The accuracy of the final testing process is not much lower than the one obtained for the validation set. It would be a good option to populate more the violent labels, but we did not want to use synthetic data in the final evaluation.

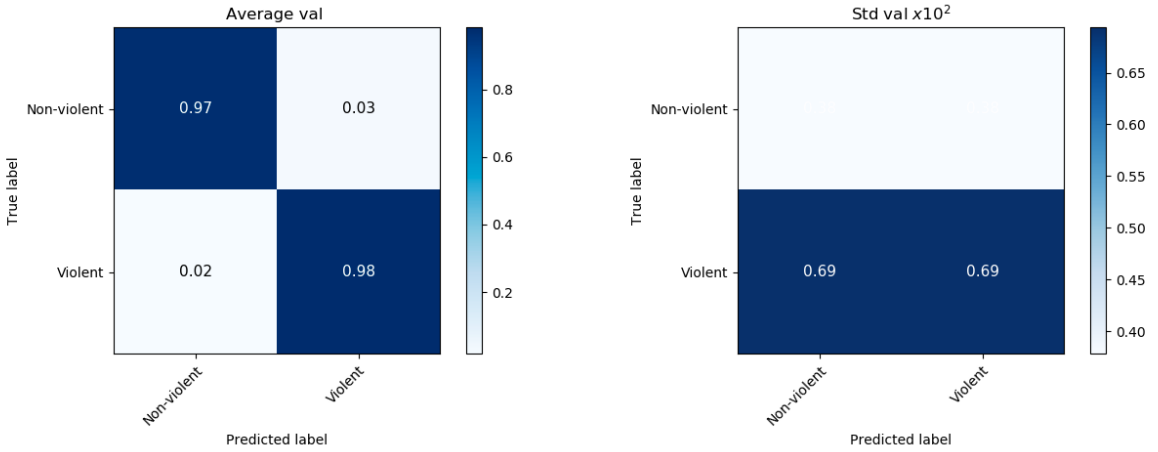
5.2.2. Implementation 2: SVM multiclass classification and SVM binary classification

The SVM multiclass model is then incorporated to the previous stage of the preprocessing. So, it predicts the multiclass probability for each value, what can also be interpreted as it is converting the data to a new feature space before the binary classification.

The results in accuracy and confusion matrices for the different evaluations can be found below. Figure 5.7 includes the average confusion matrices for train and validation sets. Then, figure 5.8 shows the confusion matrix for the test set. Finally, in figure 5.9, the accuracy values for the three sets are shown, with their perspective standard deviations.



(a) Average and standard deviation for the train set



(b) Average and standard deviation for the validation set

Fig. 5.7. Confusion matrices for SVM multiclass + SVM binary classification

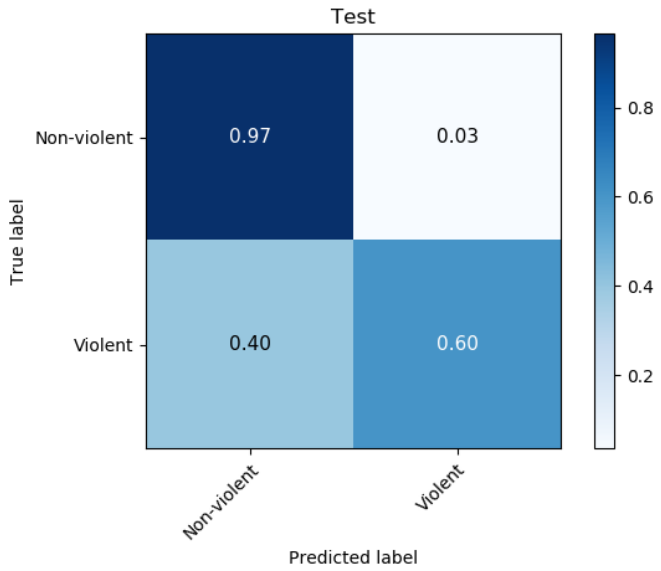


Fig. 5.8. Confusion matrix for the test set for the SVM multiclass + SVM binary classification

In this performing model it is more important to pay attention to both metrics. In the accuracy bar plot, at first, it can be interpreted that the results are really good. This can be due to the elevated number of samples for a binary classification. Considering the test value of 91%, we can see that the merit of the good performance is thank to the good classification of the non-violent observations, while the result on differentiating the violent classes is not as good since just the 60% of the labels were correctly predicted. For the train and validation sets, the results are pretty good for both types, being a 98% for both cases. Again, the fact of creating a big amount of the samples artificially for the training and validating parts is given a considerable difference in the results respect to the testing. However, this output also allows to see that with a more balanced test set the performance would be more even in the tree sets, as it happened with the multiclass task explained before, and that those observations which belong to the original embeddings show a satisfying outcome.

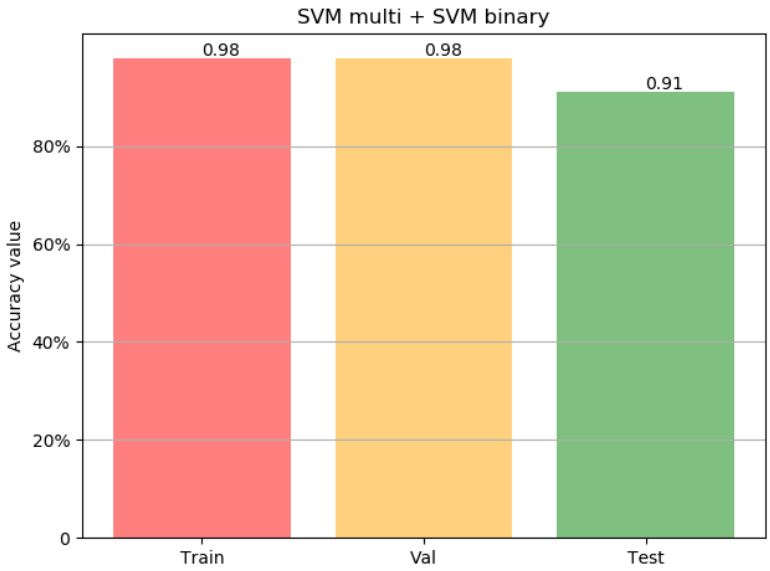


Fig. 5.9. Accuracy values for the three sets for the SVM multiclass + SVM binary classification

5.2.3. Implementation 3: LSTM for multiclass classification

For this implementation we have decided to use a network composed by a total of three layers: two LSTM and a final Fully-Connected with a softmax activation function for the classification task. The first two LSTM layers are set with a dropout of 0.05 and a recurrent dropout of 0.35. For the first one, the number of units was set to 128, in order not to change the dimensions of its output. In the second layer, it was set to 32 layers so as to reduce the dimensionality before the final prediction in the dense layer. In figure 5.10, an schema of the model is shown.

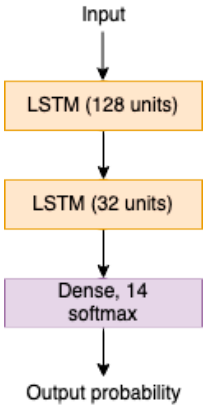


Fig. 5.10. LSTM architecture

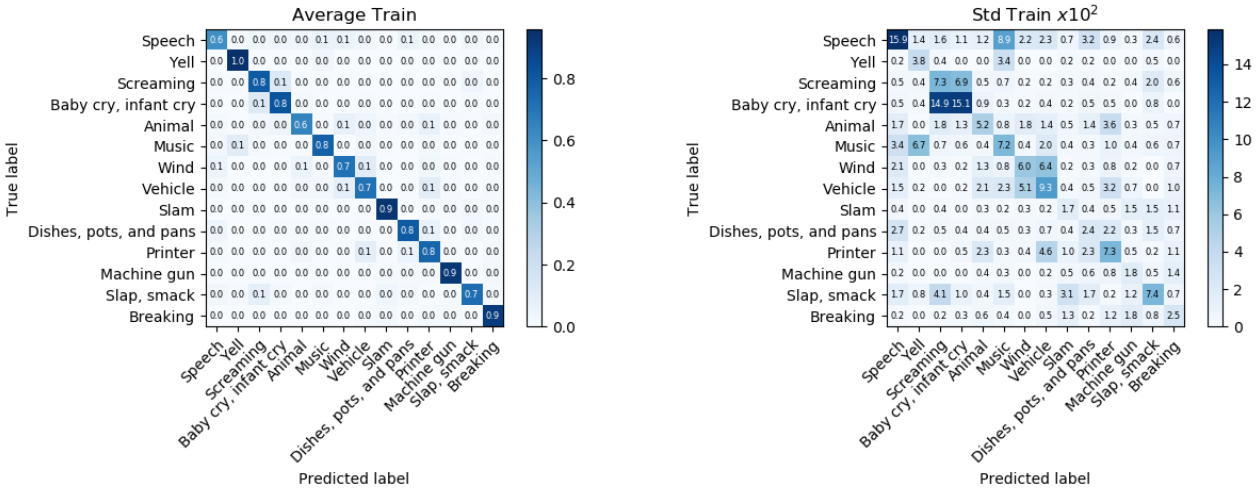
The function minimized in the process was the categorical cross-entropy, which is a common way to evaluate multiclass classification problems. A more detailed explanation of this function can be found in appendix .3. With respect to the training process, a number of 50 epochs were used and also a batch size of 32 samples [7]. Also, a Nadam optimizer was used and a learning rate of $2e^{-3}$.

remove from footnote

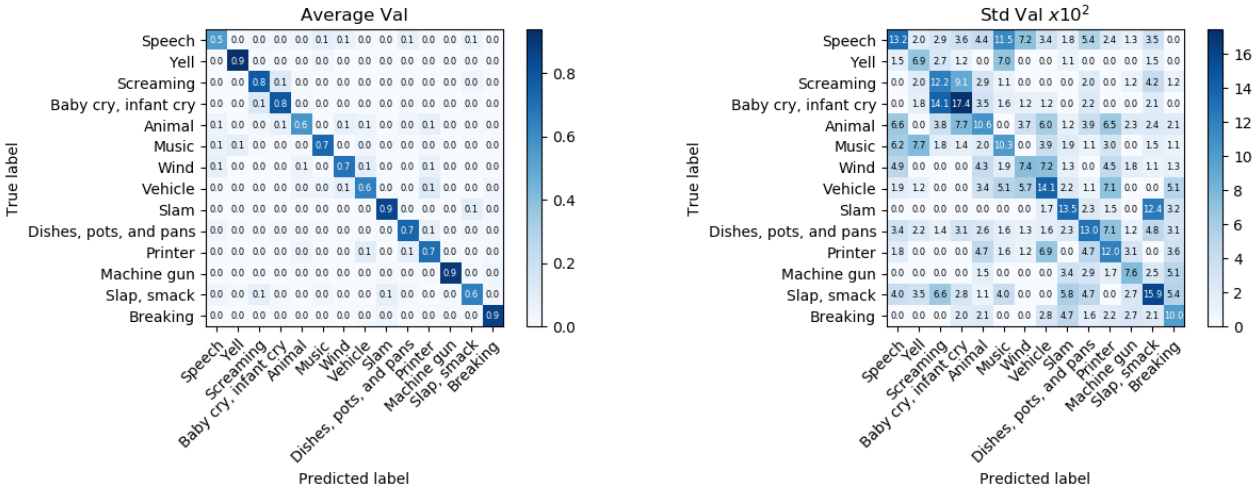
Below, the results for the classification are included. In figure 5.11, the average and

[7] When training a machine learning model, an optimization algorithm, usually SGD, is in charge of updating the internal parameters so a good performance is achieved and the loss function can be minimized. The epochs is a hyperparameter that establishes how many times the learning algorithm pass through the entire train set in the training process. The batch size is the one that defines the number of observations the algorithm works through before assigns the internal parameters an updated value [99]

standard deviation matrices for the train and validation sets are shown. Then, in figure 5.12, the confusion matrix for the testing process is included. In figure 5.13, a bar plot shows the accuracy for every set with their standard deviation values.



(a) Average and standard deviation for the train set



(b) Average and standard deviation for the validation set

Fig. 5.11. Confusion matrices for LSTM multiclass classification for train and validation sets

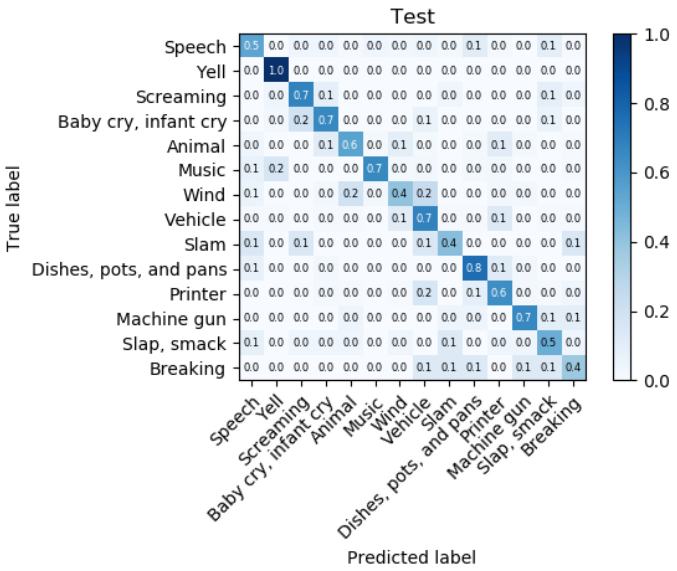


Fig. 5.12. Confusion matrix for the test set for the LSTM multiclass classification

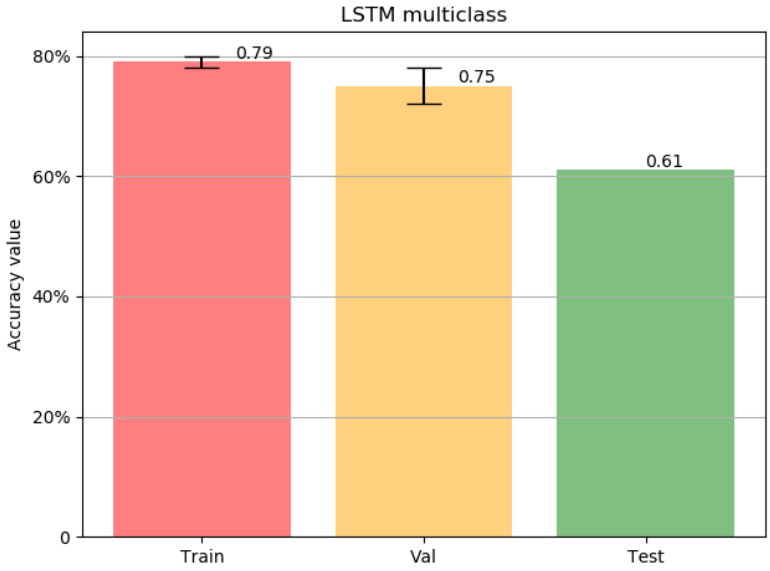


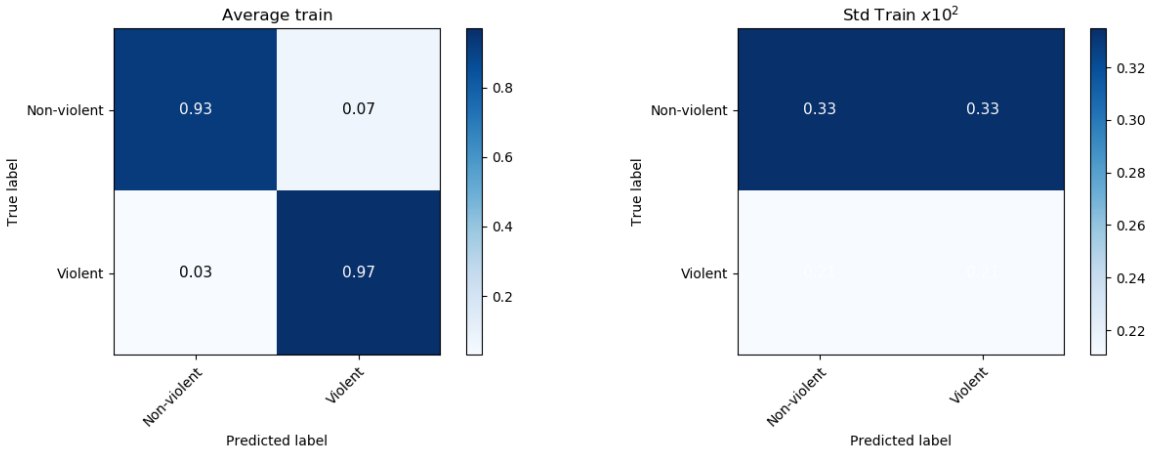
Fig. 5.13. Accuracy values for the three sets for the LSTM multiclass classification

For this model, we obtained an accuracy of $79\% \pm 1\%$ for the training set, a $75\% \pm 3\%$ for the validation and a 61% for the test. This can be due to the possible confusion regions mainly between classes such as *Screaming* and *Baby cry, infant cry*, and *Wind* and *Vehicle*, whose standard deviation values are the most emphasized. In the testing procedure, a really good performance was done for the class *Yell*. The other violent classes have a less number of correct predictions but also due to the less amount of samples that belong to these classes in the test subset. There are no signs of overfitting since the values of accuracy in the training and validation sets with respect to the test one are not really away from each other.

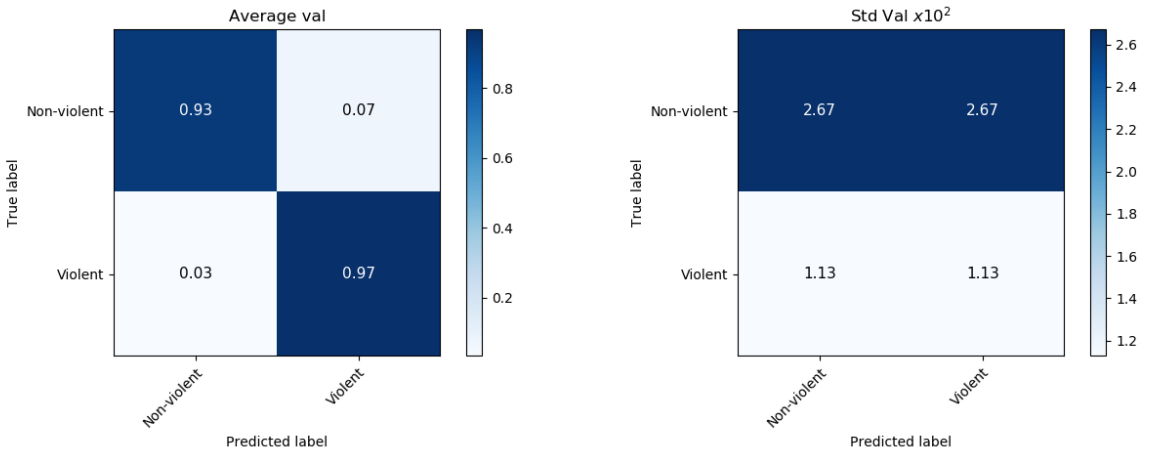
5.2.4. Implementation 4: LSTM multiclass + SVM for a final binary classification

For this case, the predicted probabilities obtained from the LSTM multiclass classifier are passed as input to the SVM for the binary classification.

The accuracy value for the different sets are included. Figure 5.14 shows the average confusion matrices for the train and validation sets. In figure 5.15, the confusion matrix for the test set is also included, and in figure 5.16, the error bar can be seen with the accuracy values for each of the sets.



(a) Average and standard deviation for the train set



(b) Average and standard deviation for the validation set

Fig. 5.14. Confusion matrices for LSTM multiclass + SVM binary classification

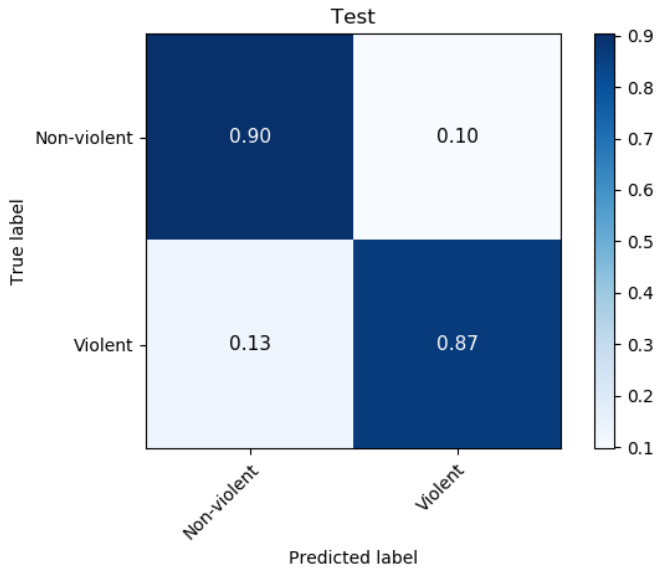


Fig. 5.15. Confusion matrix for the test set for the LSTM multiclass + SVM binary classification

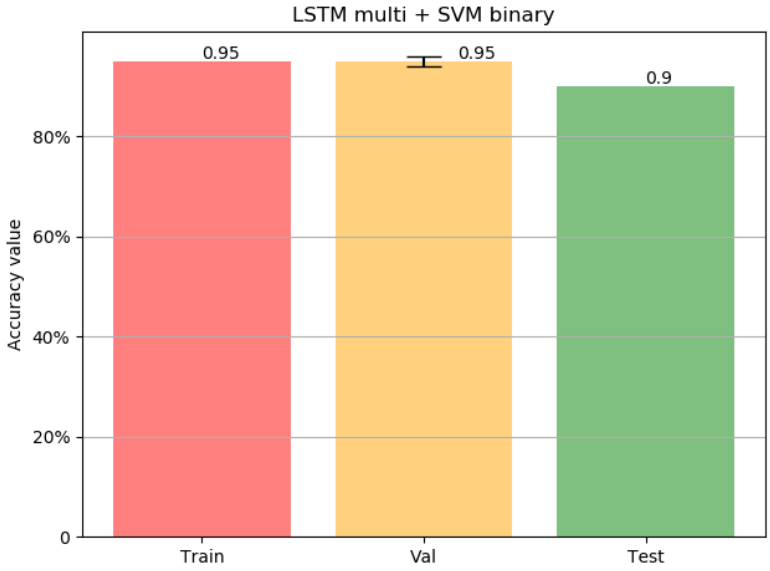


Fig. 5.16. Accuracy values for the three sets for the LSTM multiclass + SVM binary classification

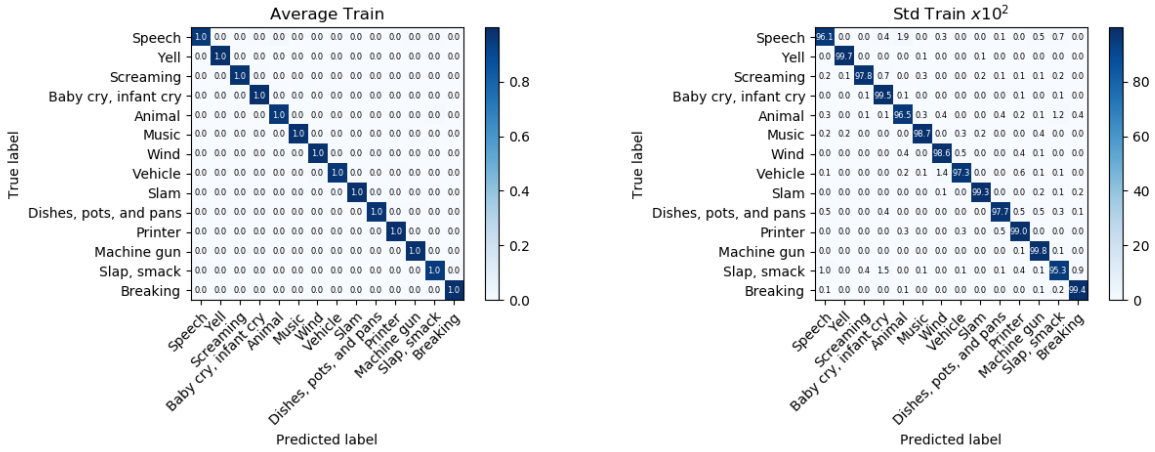
As shown in the bar plot, the values of the accuracy for train and validation are 95% and 95%±1%, respectively. The average matrices are really good as well. The no variation between them could be understood as a symptom of overfitting. However, the accuracy for testing is 90%, which is similar to the train and test results. The interesting point can be read in the confusion matrix for the test set. The classification of the violent classes is almost as good as the one for the non-violent ones, despite of the less amount of samples in these categories.

5.2.5. Implementation 5: CNN for multiclass classification

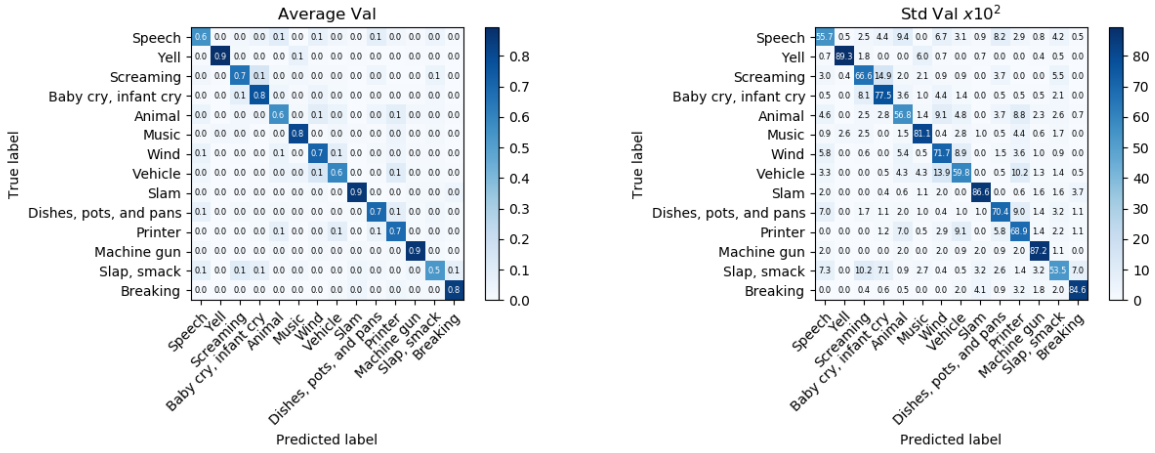
For this case, we have implemented a CNN model based on the architecture presented in figure 4.3 in subsection 4.2.3 for the small experiment to check the similarity of the embeddings from the *.tfrecord* files and the naturally audio files.

In the configuration of the network, the same hyperparameters from implementation 3 are used for the learning process: a categorical cross-entropy for the loss function, a Nadam optimizer, a number of 50 epochs and a batch size of 32 samples. We have kept the same value for this model since the number of observations is the same and the depth of the network is not very great. A learning rate of $2e^{-3}$ was also used.

In figure 5.17, the average and standard deviation matrices for the train and validation sets are shown. In figure 5.18, the confusion matrix for the test set. The bar plot in 5.19, shows the accuracy for every set with their standard deviation values.



(a) Average and standard deviation for the train set



(b) Average and standard deviation for the validation set

Fig. 5.17. Confusion matrices for CNN multiclass classification for train and validation sets

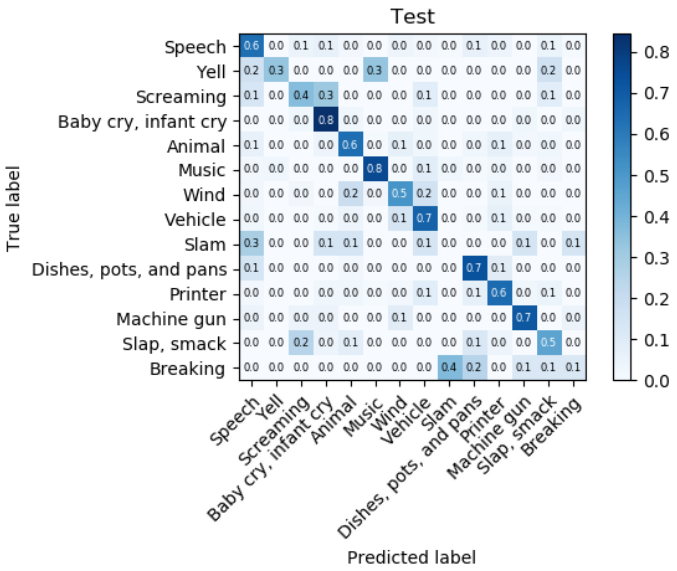


Fig. 5.18. Confusion matrix for the test set for the CNN multiclass classification

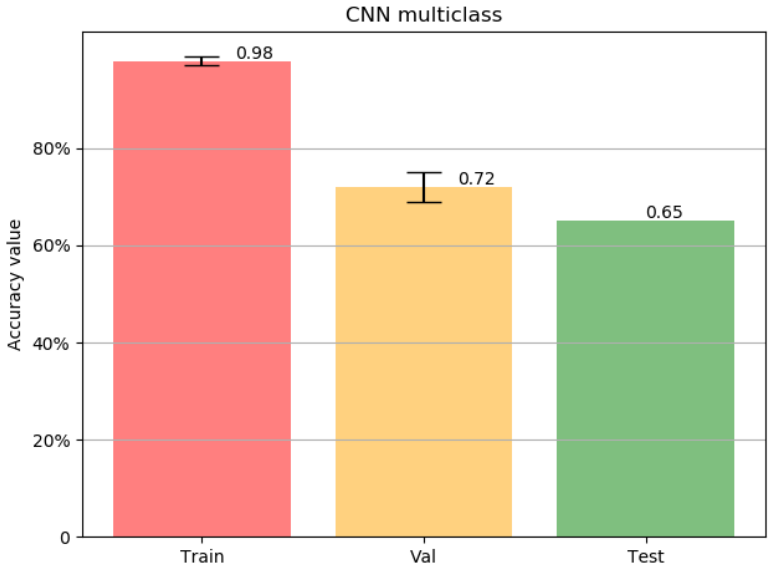


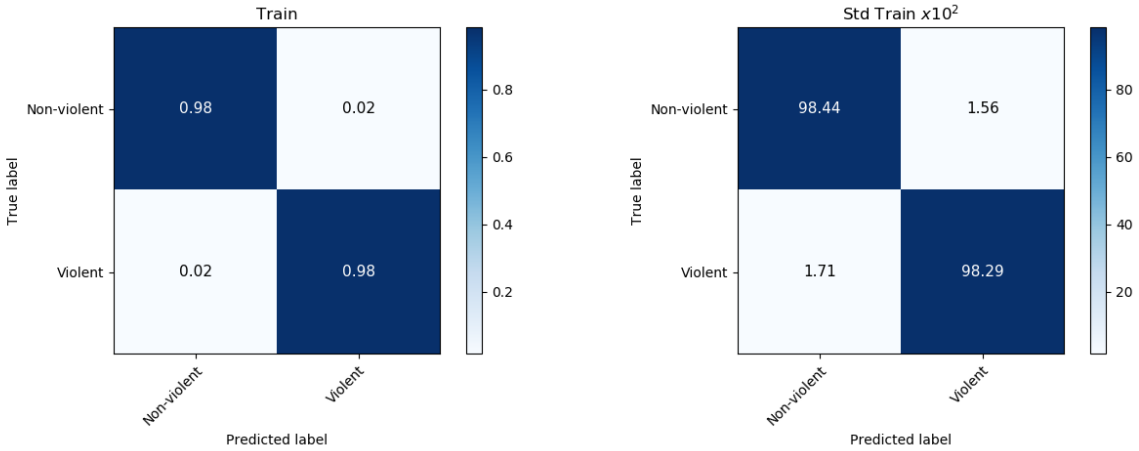
Fig. 5.19. Accuracy values for the three sets for the CNN multiclass classification

The values of the accuracy obtained for this implementation are 98% ± 1% for the train set, 72% ± 3% for validation and 65% for the test set. In the error bar in figure 5.19, the difference between the red bar for train is considerable meaningful with respect to the other two. Also, the confusion matrix for train shows a very well marked diagonal, while the other two are not that perfect. In the validation set, there are some misclassifications but still the equality among cells is present. Some confusion regions can be found again between *Screaming* and *Baby cry, infant cry* or *Wind* and *Vehicle*. In the test set, we can see that the diagonal does not follow a regular shape, being this one the worst result of the experiments.

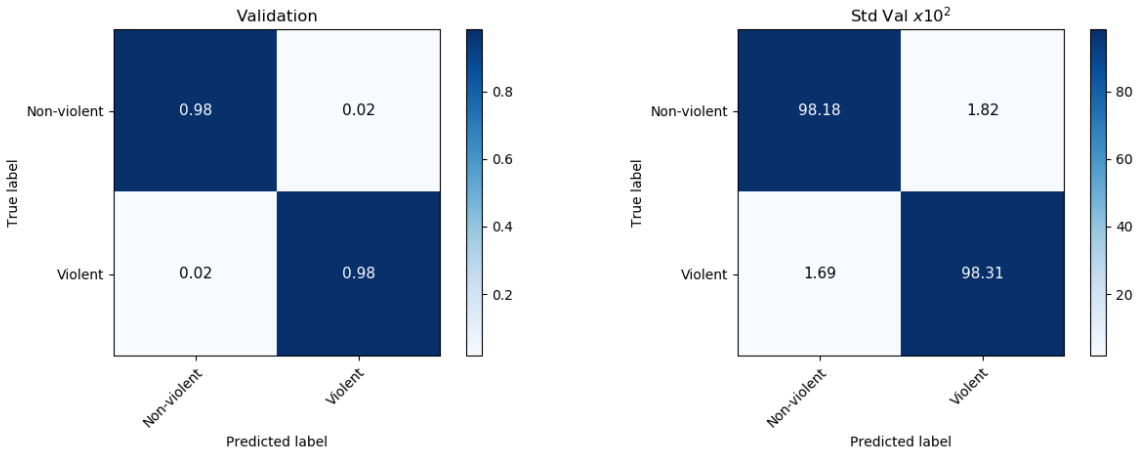
5.2.6. Implementation 6: CNN multiclass + SVM for a final binary classification

In this last approach we wanted to use the predictions from the CNN to feed the binary classifier SVM.

In figure 5.20 shows the average of the confusion matrices for the train and validation sets. In figure 5.21, it is included the confusion matrix for the test set. Finally, figure 5.22 shows the error bar in which the accuracy values for the three test can be seen.



(a) Average and standard deviation for the train set



(b) Average and standard deviation for the validation set

Fig. 5.20. Confusion matrices for CNN multiclass + SVM binary classification

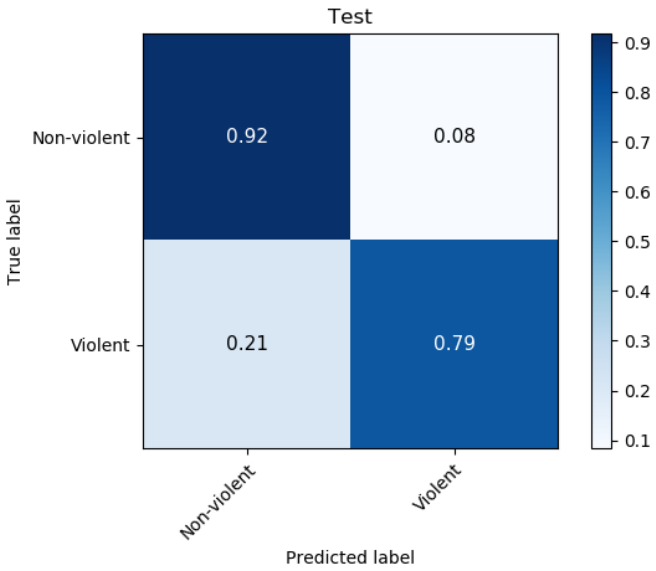


Fig. 5.21. Confusion matrix for the test set for the CNN multiclass + SVM binary classification

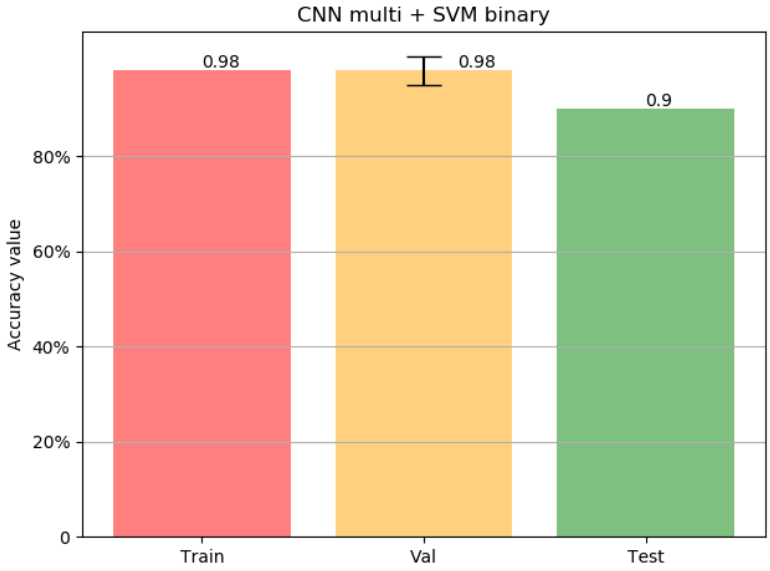


Fig. 5.22. Accuracy values for the three sets for the CNN multiclass + SVM binary classification

The values of the accuracy are 98%, 98% ± 3% and 90% for train, validation and test set, respectively. Again the results are similar to the ones obtained in the experiment 4. In this case, the classification of the violent classes are good enough as well for the test. However, as seen in experiment 5, there are plenty of misclassifications for the test set but some of them happen between classes of the same kind, so this fails do not have an effect on the binary problem. Also, others have been **mixed** with categories from the other group, this is would be the reason of the lower output.

5.3. Comparison and conclusion

The idea behind the different implementations was to check the usefulness of the embeddings extracted with VGGish for a violent/non-violent classification problem. We wanted to study the performance of different learning techniques to see how this type of data worked with them. To do so, we first ran a typical algorithm for classification that do not involve Neural Network, as it is the SVM. Then, we thought it could be a good idea to take put in practice neural models with different core concepts to exploit the nature of our data. The dimensions of our data let us to make a try on the CNN, while the temporal character of an audio instance makes the LSTM a possible approach. In this section, we are going to compare the six different results dividing the explanation according to the type classification: multiclass or binary. For the multiclass problem, the results are shown again in table 5.3

	Train	**Validation**	**Test**
SVM	92%	86%	72%
LSTM	79% ± 1%	75% ± 3%	61%
CNN	98% ± 2%	72% ± 3%	65%

Table 5.3. ACCURACY RESULTS FOR THE THREE DIFFERENT ALGORITHMS AND THE THREE SETS

Initially, the best performance for this case is the one done by with SVM. It is expectable that this algorithm achieves a good result due to the way it works. Basically, it expresses the different data sample in a feature space in which the optimal frontier is set, as explained in subsection 4.3.2. However, this one is not totally comparable with the other two LSTM and CNN, since the number of observations to feed the model is much greater because of the conversions explained in the subsection 5.1. This way, the method has more examples to learn how the different classes are distributed.

For the CNN approach, the results are maybe the least satisfactory. The system presents a clear overfitting due to the big difference of the results for within the three sets. It is learning the data in the training stage and not performing a not being able to generalize so as to perform a good enough classification for validation and test. Also, considering the fails in figure 5.18, the mismatches are the greatest within the three experiments.

In the LSTM approach, the results are numerically the worst but also the ones that make more sense. The system does not present an overfitting since the accuracies for validation and test are similar to the one for train. Also, the errors that can be read in the confusion matrix in figure 5.15 are understandable since the audio data for those classes involved in the confusion regions may look similar. For example, a recording that belong to *Baby cry, infant cry* is logical that is similar to one from *Screaming*. Same explanation could be use for *Wind* and *Vehicle*, since they usually have sounds that belong to scenes

in a rush.

Let's look at some violent categories to check their performance.

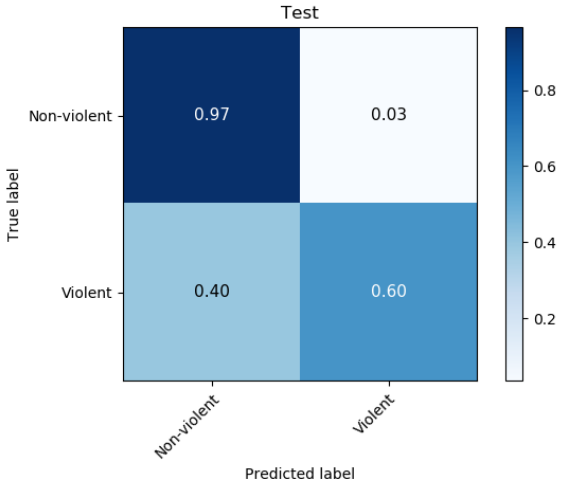
- The error when predicting *Music* for *Yell* samples that appears in SVM and CNN results, is not a problem in the classification of the test set for the LSTM. In fact, all the *Yell* samples belong to the true positive cell.
- The confusion region between *Screaming* and *Baby cry, infant cry* comes out in all matrices. This makes sense, since the semantic of these two type of audio events is really similar.
- The predictions for *Slam* are more or less the same between the SVM and LSTM, obtaining a true positive of 40% and a distribution of the false negatives along all the other categories. For the CNN, this class is a completely fail being most of it classified as *Speech*.
- In the case of *Machine gun* it has a pretty well performance for all the three models, getting a 70% for LSTM and CNN and a 60% for the SVM of the samples as true positives. The rest of them are shared uniformly along all the other classes.
- In the case of *Slap, smack* is predicted in a similar way for all the models as well, achieving a 50% of the samples right classified.
- The class *Breaking* has given similar outputs in SVM and LSTM being this last one a 10% better. In the case of the CNN model, it was mainly predicted as *Slam*, being the true positive cell with just a 10%.

The conclusion we can extract from this comparison is that the model that shows a much better result in the sense of violent categories is the LSTM. However, when looking at the non-violent ones, the SVM makes a better performance, achieving the highest value in the true positive place for *Music* with a 90%. The best system to keep working on the field may be the LSTM, since the sequential property of an audio file can be usable. Also, it is the one that best adapts the original features to the binary problem, as it will be explained below.

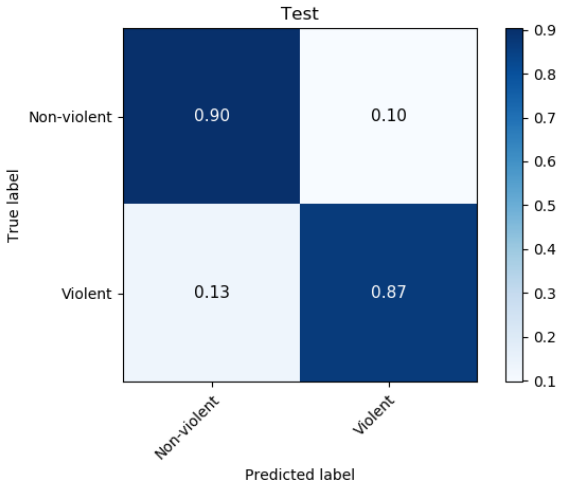
For the binary problem, the results are included in table 5.4. Also the confusion matrices for test are shown again in figure 5.23, since they are really relevant for the final explanation.

	Train	Validation	Test
SVM	98%	98%	91%
LSTM	95%	95% ± 1%	90%
CNN	98%	98% ± 3%	90%

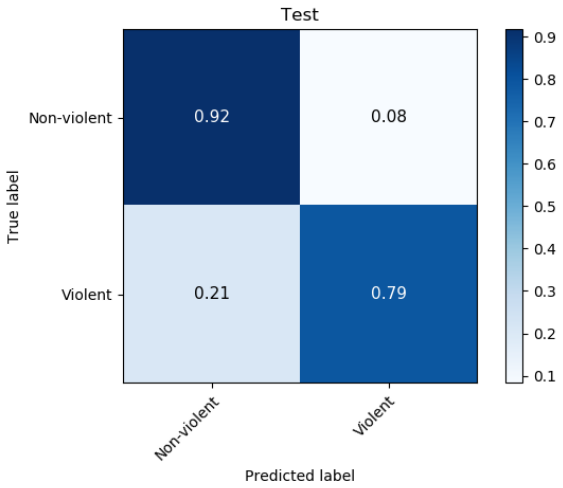
Table 5.4. ACCURACY RESULTS FOR THE THREE DIFFERENT ALGORITHMS AND THE THREE SETS



(a) SVM multiclass + SVM binary



(b) LSTM multiclass + SVM binary



(c) CNN multiclass + SVM binary

The actual solution of the problem would be this binary classification that allows us to distinguish between non-violent and violent events. The results apparently are really good if we look at the accuracy values, but the interpretability is much more clear when reading the confusion matrices.

In order to interpret the result for the SVM, we can read the confusion matrix in figure 5.5 to check how the multiclass classification performed and related to the output obtained from the binary classifier. For this case, certain classes have a wide range of false negative in non-violent categories, such as *Yell* whose has pretty fails place in *Music*. Also, a couple of categories are predicted as *Speech* and some others as *Animal*. If we interpret this fails in a binary context, they can be resumed by saying that violent labels are predicted as non-violent. This fact is reflected in the only 60% of true positives for the violent label in figure 5.23 *(a)*.

In the CNN approach, the results are numerically smaller but the classification of the violent instances are much better. Almost a 87% of these samples were rightly classified. Again, if we look at figure 5.18, some classes are confused within the two different binary labels. A 30% of instances from *Slam* have been wrongly identified as *Speech*. The same happens with *Yell* and *Music* as in the case of SVM. So, these can affect the output of this binary approach and the result of the test confusion matrix. There is also a big mistake in predicting *Slam* for samples that belong to *Breaking*, but this fact does not involve errors in the binary classification.

Finally, the LSTM is the method for which a better result is obtained in this problem. In the confusion matrix in figure 5.12, as explained above, we can see some misclassifications but they happen between data of the same type. In a real application this would be a more reasonable result, in order to detect a violent situation without depending too much on the exact event occurred just considering if it is violent or not. This is why the percentage of the true positive samples for the violent class is the best for this model with a 82%. We can say that the best approach for the final experiment of the work is the one that involves LSTM as multiclass classifier.

Esta parte me costó hacerla porque no sabía muy bien por donde tirar. Al final opté por

ACRONYMS

.csv Comma-separated values. 19, 32, 33

.wav Waveform Audio File Format. 32

AED Acoustic Event Detection. 6, 8, 15

AED/C Acoustic Event Detection and Classification. vii, 3

ANN Artificial Neural Network. vii, 8, 19, 20, 24, 41

ASC Acoustic Scene Classification. vii, 3, 6, 7, 12

BAA Broad Agency Announcement. 10

BOF Bag-of-frames. 6, 7

CASA Computational Acoustic Scenes Analysis. 3

CHIL Computers in the Human Interaction Loop. 15

CNN Convolutional Neural Network. vii, viii, 8, 9, 19, 20, 21, 25, 27, 29, 34, 46, 48, 60, 62, 64, 65, 66, 67

ConvNet Convolutional Network. 20, 21, 22, 24, 25, 26

CV Computer Vision. 3, 8, 25

D Dimensions. 21, 22, 23

DARPA Defense Advanced Research Projects Agency. 10

DFT Discrete Fourier Transform.

DNN Deep Neural Network. 9, 12, 20

FC Fully-Connected. 8, 20, 21, 22, 24, 25, 26, 27, 35, 55

FFT Fast Fourier Transform. 5,

FT Fourier Transform.

GMM Gaussian Mixture Models. 6, 8

HMM Hidden Markov Models. 7, 8

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 25, 26

IPTO Information Processing Technology Office. 10

IPV Intimate Partner Violence. 13

k-Fold k-Fold Cross Validation. viii, 48, 50,

L^3 Look, Listen and Learn. 9

LLD low-level descriptors. 4, 5, 6, 7

LRN Local Response Normalisation. 22, 26

LSTM Long Short Term Memory. viii, 8, 9, 41, 43, 46, 48, 55, 58, 64, 65, 66, 67

MED Multimedia Event Detection. 6

MFCC Mel-frequency Cepstrum Coefficients. 5, 6, 8,

MID Machine ID. 17, 18

MLP multi-layer perceptron. 20

Nadam Nesterov-accelerated Adaptive Moment Estimation. 55, 60

NN Neural Network. 31, 36, 50, 64

PCA Principal Components Analysis. 29, 36

RBF Radial Basis function. 41, 50

ReLU Rectified Linear Unit. 22, 26, 35

RGB red, green and blue. 21, 26

RNN Recurrent Neural Network. viii, 8, 41, 42, 43, 44

SC Spectral Centroid. 5

SF Spectral Flux. 5, 6

SGD Stochatic Gradient Descend. 55

SMOTE Synthetic Minority Over-sampling Technique. viii, 39, 47

SNE Stochastic Neighbor Embedding. 37

STFT Short-Time Fourier Transform. 5, 28,

SVC C-Support Vector Classification. 41

SVM Support Vector Machine. viii, x, 7, 8, 9, 40, 41, 46, 47, 48, 50, 52, 58, 62, 64, 65, 67

tanh Hyperbolic tangent. 44, 45

t-SNE t-distributed Stochastic Neighbor Embedding. 34, 37

TF TensorFlow. 19, 35

UC3M Universidad Carlos III de Madrid. 13

URL Uniform Resource Locator. 32

VGG Visual Geometry Group. vii, 9, 11, 19, 25, 26, 27, 29, 30, 31, 33, 64

ZCR Zero Crossing Rate. 5

BIBLIOGRAPHY

[1] D. Barchiesi, D. D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic Scene Classification: Classifying environments from the sounds they produce", *IEEE Signal Processing Magazine*, 2015. DOI: 10.1109/MSP.2014.2326181.

[2] D. Dubois, C. Guastavino, and M. Raimbault, "A cognitive approach to urban soundscapes: Using verbal data to access everyday life auditory categories", *Acta Acustica united with Acustica*, 2006.

[3] D. Wang and G. J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*. 2006. DOI: 10.1109/9780470043387.

[4] A. J. Eronen *et al.*, "Audio-based context recognition", in *IEEE Transactions on Audio, Speech and Language Processing*, 2006. DOI: 10.1109/TSA.2005.854103.

[5] M. Bahoura, "Pattern recognition methods applied to respiratory sounds classification into normal and wheeze classes", *Computers in Biology and Medicine*, 2009. DOI: 10.1016/j.compbiomed.2009.06.011.

[6] D. Van Nort, P. Oliveros, and J. Braasch, "Electro/acoustic improvisation and deeply listening machines", *Journal of New Music Research*, vol. 42, no. 4, pp. 303–324, 2013.

[7] A. Temko *et al.*, "Acoustic Event Detection and Classification", in *Computers in the Human Interaction Loop*, 2009, ch. Part II, 7. DOI: 10.1007/978-1-84882-054-8_7.

[8] A. Temko *et al.*, "CLEAR evaluation of acoustic event detection and classification systems", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007. DOI: 10.1007/978-3-540-69568-4_29.

[9] E. S. Sazonov *et al.*, "Automatic detection of swallowing events by acoustical means for applications of monitoring of ingestive behavior", *IEEE Transactions on Biomedical Engineering*, 2010. DOI: 10.1109/TBME.2009.2033037.

[10] I. Potamitis, S. Ntalampiras, O. Jahn, and K. Riede, "Automatic bird sound detection in long real-field recordings: Applications and tools", *Applied Acoustics*, 2014. DOI: 10.1016/j.apacoust.2014.01.001.

[11] Y. Wang, L. Neves, and F. Metze, "Audio-based multimedia event detection using deep recurrent neural networks", in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2016. DOI: 10.1109/ICASSP.2016.7472176.

[12] T. Giannakopoulos and A. Pikrakis, *Introduction to Audio Analysis: A MATLAB Approach*. 2014. DOI: 10.1016/C2012-0-03524-7.

[13] X. Amatriain, "An Object-Oriented Metamodel for Digital Signal Processing with a focus on Audio and Music", 2004.

[14] D. Marr, "Vision: a computational investigation into the human representation and processing of visual information.", *Vision: a computational investigation into the human representation and processing of visual information.*, 1982. DOI: 10.1016/0022-2496(83)90030-5.

[15] T. Giannakopoulos, D. Kosmopoulos, A. Aristidou, and S. Theodoridis, "Violence content classification using audio features", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006. DOI: 10.1007/11752912_55.

[16] J. García-Gómez, M. Bautista-Durán, R. Gil-Pita, I. Mohino-Herranz, and M. Rosa-Zurera, "Violence detection in real environments for smart cities", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016. DOI: 10.1007/978-3-319-48799-1_52.

[17] O. BÜYÜK and M. L. ARSLAN, "Combination of Long-Term and Short-Term Features for Age Identification from Voice", 2018.

[18] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and Classification of Acoustic Scenes and Events", *IEEE Transactions on Multimedia*, 2015. DOI: 10.1109/TMM.2015.2428998.

[19] J. R. Uijlings, A. W. Smeulders, and R. J. Scha, "Real-time Bag of Words, approximately", in *CIVR 2009 - Proceedings of the ACM International Conference on Image and Video Retrieval*, 2009. DOI: 10.1145/1646396.1646405.

[20] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music", *The Journal of the Acoustical Society of America*, 2007. DOI: 10.1121/1.2750160.

[21] J.-J. Aucouturier and B. Defreville, "Judging the similarity of soundscapes does not require categorization: Evidence from spliced stimuli", *The Journal of the Acoustical Society of America*, 2009. DOI: 10.1121/1.3083232.

[22] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings", in *European Signal Processing Conference*, 2010.

[23] T. Grill, "Constructing high-level perceptual audio descriptors for textural sounds", in *Proceedings of the 9th Sound and Music Computing Conference, SMC 2012*, 2012.

[24] F. Pachet and P. Roy, “Analytical features: A knowledge-based approach to audio feature generation”, *Eurasip Journal on Audio, Speech, and Music Processing*, 2009. DOI: 10.1155/2009/153017.

[25] D. Bogdanov, J. Serrà, N. Wack, P. Herrera, and X. Serra, “Unifying low-level and high-level music similarity measures”, *IEEE Transactions on Multimedia*, 2011. DOI: 10.1109/TMM.2011.2125784.

[26] H. Jiang, J. Bai, S. Zhang, and B. Xu, “SVM-based audio scene classification”, in *Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, IEEE NLP-KE'05*, 2005. DOI: 10.1109/NLPKE.2005.1598721.

[27] J. T. Geiger, B. Schuller, and G. Rigoll, “Large-scale audio feature extraction and SVM for acoustic scene classification”, in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013. DOI: 10.1109/WASPAA.2013.6701857.

[28] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, “A survey of audio-based music classification and annotation”, *IEEE Transactions on Multimedia*, 2011. DOI: 10.1109/TMM.2010.2098858.

[29] X. Zhuang, X. Zhou, M. A. Hasegawa-Johnson, and T. S. Huang, “Real-world acoustic event detection”, *Pattern Recognition Letters*, 2010. DOI: 10.1016/j.patrec.2010.02.005.

[30] Z. Ren, Q. Kong, K. Qian, M. D. Plumbley, and B. W. Schuller, “ATTENTION-BASED CONVOLUTIONAL NEURAL NETWORKS FOR ACOUSTIC SCENE CLASSIFICATION”, 2018.

[31] J. Cramer, H. H. Wu, J. Salamon, and J. P. Bello, “Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings”, in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2019. DOI: 10.1109/ICASSP.2019.8682475.

[32] Y. Aytar, C. Vondrick, and A. Torralba, “SoundNet: Learning sound representations from unlabeled video”, in *Advances in Neural Information Processing Systems*, 2016. arXiv: 1610.09001.

[33] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. Lecun, “Unsupervised learning of sparse features for scalable audio classification”, in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*, 2011.

[34] J. Nam, J. Herrera, M. Slaney, and J. Smith, “Learning sparse feature representations for music annotation and retrieval”, in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, 2012.

[35] K. Lee, Z. Hyung, and J. Nam, "Acoustic scene classification using sparse feature learning and event-based pooling", in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013. DOI: 10.1109/WASPAA.2013.6701893.

[36] S. J. Pan and Q. Yang, *A survey on transfer learning*, 2010. DOI: 10.1109/TKDE.2009.191.

[37] D. Sarkar, "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning", 2018.

[38] S. Ruder, "Transfer Learning - Machine Learning's Next Frontier", 2017.

[39] E. G. Krug, J. A. Mercy, L. L. Dahlberg, and A. B. Zwi, "The world report on violence and health", *Lancet*, 2002. DOI: 10.1016/S0140-6736(02)11133-0.

[40] C. H. Demarty *et al.*, "The MediaEval 2013 affect task: Violent Scenes Detection", in *CEUR Workshop Proceedings*, 2013.

[41] T. Giannakopoulos, A. Makris, D. Kosmopoulos, S. Perantonis, and S. Theodoridis, "Audio-visual fusion for detecting violent scenes in videos", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010. DOI: 10.1007/978-3-642-12842-4_13.

[42] A. Ali and N. Senan, "Violence video classification performance using deep neural networks", *Advances in Intelligent Systems and Computing*, vol. 700, pp. 225–233, 2018. DOI: 10.1007/978-3-319-72550-5_22.

[43] T. W. Chua, K. Leman, and F. Gao, "Hierarchical audio-visual surveillance for passenger elevators", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. DOI: 10.1007/978-3-319-04117-9_5.

[44] M. Bautista-Duran *et al.*, "Acoustic detection of violence in real and fictional environments", in *ICPRAM 2017 - Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*, 2017. DOI: 10.5220/0006195004560462.

[45] United Nations, *No Title*, 1989.

[46] WHO. Department of Reproductive Health Research. London School of Hygiene and Tropical Medicine. South African Medical Research Council., *WHO | Global and regional estimates of violence against women*. 2013.

[47] European Union Agency for Fundamental Rights, *Violence against women : An EU-wide survey*. 2014. DOI: 10.2811/62230.

[48] L. Heise, M. Ellsberg, and M. Gottemoeller, *Ending violence against women.* 1999. DOI: 10.4324/9780429269516-5.

[49] K. Beyer, A. B. Wallis, and L. K. Hamberger, "Neighborhood Environment and Intimate Partner Violence: A Systematic Review", *Trauma, Violence, and Abuse*, 2015. DOI: 10.1177/1524838013515758.

[50] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello, "Scaper: A library for soundscape synthesis and augmentation", in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2017. DOI: 10.1109/WASPAA.2017.8170052.

[51] V. Mapell, *UPC-TALP database of isolated meeting-room acoustic events*, 2012. [Online]. Available: http://catalog.elra.info/en-us/repository/browse/ELRA-S0268/.

[52] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Reliable detection of audio events in highly noisy environments", *Pattern Recognition Letters*, 2015. DOI: 10.1016/j.patrec.2015.06.026.

[53] E. Fagerlund and A. Hiltunen, *TUT Rare sound events*, 2017.

[54] D. Stowell and E. Benetos, *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events*, 2013.

[55] E. Cakir and T. Heittola, *TUT-SED Synthetic*, 2016.

[56] C. H. Demarty, C. Penet, M. Soleymani, and G. Gravier, "VSD, a public dataset for the detection of violent scenes in movies: design, annotation, analysis and evaluation", *Multimedia Tools and Applications*, 2015. DOI: 10.1007/s11042-014-1984-4.

[57] J. F. Gemmeke *et al.*, "Audio Set: An ontology and human-labeled dataset for audio events", in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017. DOI: 10.1109/ICASSP.2017.7952261.

[58] E. Fonseca *et al.*, "Freesound datasets: A platform for the creation of open audio datasets", in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, 2017.

[59] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora", 1992. DOI: 10.3115/992133.992154.

[60] A. Singhal, *Introducing the Knowledge Graph: things, not strings*, 2012.

[61] Sound Understanding group, *AudioSet*, 2017.

[62] GoogleResearch, "TensorFlow: Large-scale machine learning on heterogeneous systems", *Google Research*, 2015. arXiv: arXiv:1603.04467v2.

[63] S. Hershey *et al.*, "CNN architectures for large-scale audio classification", in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017. DOI: 10.1109/ICASSP.2017.7952132. arXiv: 1609.09430.

[64] S. Abu-El-Haija *et al.*, "YouTube-8M: A Large-Scale Video Classification Benchmark", *CoRR*, vol. abs/1609.0, 2016. [Online]. Available: http://arxiv.org/abs/1609.08675.

[65] S. J. Kwon, *Artificial neural networks*. 2011. DOI: 10.4324/9781315154282-3.

[66] L. Deng, *A tutorial survey of architectures, algorithms, and applications for deep learning*, 2014. DOI: 10.1017/atsip.2013.9.

[67] W. Di, A. Bhardwaj, and W. Jianing, *Deep Learning Essentials*. 2018.

[68] A. Karpathy, *CS231n Convolutional Neural Networks for Visual Recognition*, 2016. [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[69] S. Saha, *No A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, 2018.

[70] A. Anwar, *Difference between Local Response Normalization and Batch Normalization*, 2019. [Online]. Available: https://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization-272308c034ac.

[71] Keras, *Convolutional Layers*.

[72] H. Mahmood, *The Softmax Function, Simplified*, 2018.

[73] T. Hinz, P. Barros, and S. Wermter, "The Effects of Regularization on Learning Facial Expressions with Convolutional Neural Networks", 2016, pp. 80–87. DOI: 10.1007/978-3-319-44781-0_10.

[74] K. J. Piczak, "Environmental sound classification with convolutional neural networks", in *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, 2015. DOI: 10.1109/MLSP.2015.7324337.

[75] A. Kumar and B. Raj, "Deep CNN Framework for Audio Event Recognition using Weakly Labeled Web Data", 2017.

[76] ImageNet, *Results for ILSVRC2014*, 2014.

[77] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. arXiv: 1409.1556.

[78] D. Ellis, S. Hershey, A. Jansen, and M. Plakal, *VGGish code*, 2017. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/audioset.

[79] H. Abdi and L. J. Williams, *Principal component analysis*, 2010. DOI: 10.1002/wics.101.

[80] M. Levoy, K. Dektar, and A. Adams, *Spatial Convolution*, 2012. [Online]. Available: https://graphics.stanford.edu/courses/cs178/applets/convolution.html.

[81] M. Rampurawala, *Classification with TensorFlow and Dense Neural Networks*, 2019.

[82] H. K. Jabbar and R. Z. Khan, "Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study)", 2015. DOI: 10.3850/978-981-09-5247-1_017.

[83] S. Kaski and J. Peltonen, "Dimensionality reduction for data visualization", *IEEE Signal Processing Magazine*, 2011. DOI: 10.1109/MSP.2010.940003.

[84] J. Amat Rodrigo, *Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE*, 2017.

[85] G. Hinton and S. Roweis, "Stochastic neighbor embedding", in *Advances in Neural Information Processing Systems*, 2003.

[86] L. Van Der Maaten and G. Hinton, "Visualizing data using t-SNE", *Journal of Machine Learning Research*, 2008.

[87] M. Wattenberg, F. Viegas, and I. Johnson, "How to Use t-SNE Effectively", *Distill*, 2016. DOI: 10.23915/distill.00002. [Online]. Available: http://distill.pub/2016/misread-tsne.

[88] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique", *Journal of Artificial Intelligence Research*, 2002. DOI: 10.1613/jair.953. arXiv: 1106.1813.

[89] J. Browniee, *SMOTE Oversampling for Imbalanced Classification with Python*, 2020.

[90] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data", *BMC Bioinformatics*, 2013. DOI: 10.1186/1471-2105-14-106.

[91] Y. Xie, Y. Liu, and Q. Fu, "Imbalanced Data Sets Classification Based on SVM for Sand-Dust Storm Warning", *Discrete Dynamics in Nature and Society*, 2015. DOI: 10.1155/2015/562724.

[92] S. Bhattacharyya, *Support Vector Machine: Kernel Trick; Mercer's Theorem*, 2018. [Online]. Available: https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d.

[93] G. Drakos, *Support Vector Machine vs Logistic Regression*, 2018.

[94] C. Olah, *Understanding LSTM Networks*, 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[95] SuperDataScience Team, *Recurrent Neural Networks (RNN) - The Vanishing Gradient Problem*, 2018. [Online]. Available: https://www.superdatascience.com/blogs/recurrent-neural-networks-rnn-the-vanishing-gradient-problem/.

[96] M. Nguyen, *Illustrated Guide to LSTM's and GRU's: A step by step explanation*, 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[97] A. Kaw, *Holistic Numerical Methods*. [Online]. Available: https://nm.mathforcollege.com/blog/.

[98] G. L. Prajapati and A. Patle, "On performing classification using SVM with radial basis and polynomial kernel functions", in *Proceedings - 3rd International Conference on Emerging Trends in Engineering and Technology, ICETET 2010*, 2010. DOI: 10.1109/ICETET.2010.134.

[99] J. Browniee, *Difference Between a Batch and an Epoch in a Neural Network*, 2018. [Online]. Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/.

[100] Scikit-learn, *Metrics and scoring: quantifying the quality of predictions*. [Online]. Available: https://scikit-learn.org/stable/modules/model%7B%5C_%7Devaluation.html.

[101] A. Mishra, *Metrics to Evaluate your Machine Learning Algorithm*, 2018. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[102] F. Krüger, "Activity, Context, and Plan Recognition with Computational Causal Behaviour Models", *ResearchGate*, 2018.

[103] J. Browniee, *A Gentle Introduction to k-fold Cross-Validation*, 2018. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/.

[104] S. M, *Why and how to Cross Validate a Model?*, 2018. [Online]. Available: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f.

[105] Scikit-learn, *Cross-validation: evaluating estimator performance*.

[106] ML Glossary, *Loss Functions*, 2017. [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss%7B%5C_%7Dfunctions.html.

[107] N. Laskaris, *How to apply machine learning and deep learning methods to audio analysis*, 2019.

[108] Y. Lei, *Intelligent fault diagnosis and remaining useful life prediction of rotating machinery*. 2016, p. 37.

[109] N. Kehtarnavaz, *Digital Signal Processing System Design*. 2008, pp. 175–196.

[110] R. X. Gao and R. Yan, "Non-stationary signal processing for bearing health monitoring", *International Journal of Manufacturing Research*, 2006. DOI: 10.1504/IJMR.2006.010701.

[111] D. Gartzman, *Getting to Know the Mel Spectrogram*, 2019. [Online]. Available: https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0.

[112] L. L. Beranek and W. A. Rosenblith, " Acoustic Measurements ", *Physics Today*, 1950. DOI: 10.1063/1.3066788.

APPENDIX X

.1. Metrics

A fundamental part of a machine learning project consists of checking the performance. There are plenty of metrics to carry out this evaluation and the results will look in one way or another depending on the method utilized. The following two are the most used in this project

.1.1. Classification Accuracy

This is a technique commonly used and it is usually referred to as just accuracy. It can be defined as the relation between the amount of right predictions and the total number on input instances [100].

$$acc = \frac{Number\ of\ incorrect\ predictions}{Number\ of\ total\ input\ instances}$$

This metric best works when dealing with a balanced dataset, i.e., the same of number of samples per class. If the problem is addressed with unbalanced data, then the accuracy value could be a higher value due to predict all the instances belong to the major class. For example, if 90% of the data are part of the same class A and all the predictions results are this class, then the accuracy value will be 90%, which apparently is a satisfying output, even though we are misclassifying all the samples from class B [101].

.1.2. Confusion matrix

As it own name describes, the output of this type of metric consists of a matrix which shows a complete evaluation of the model. By definition, an entry i, j of the matrix denotes the amount of observations that belong to group i but are predicted as group j [100]. For example, considering a binary classification problem in which there are two classes, YES

n = 165	Predicted: NO	Predicted: YES
Label: NO	50	10
Label: YES	5	100

Fig. 24. Example of confusion matrix

and NO, for a test set composed by 165 samples, the matrix included in figure 24 is obtained.

There are four groups that can be extracted from this matrix: True positives, the samples that are predicted as YES and that is in fact their true label, True Negatives, those cases that were predicted as NO and they are originally labelled as NO, False Positives, in which the predicted label is YES but they are actually negative, and False Negatives, those in which the predicted label is NO when their original label is YES.

This metric an the one explained before, accuracy, can be related by taking the diagonal of the matrix and computing the next operation:

$$acc = \frac{TruePositives + \ FalseNegatives}{Total\ number\ of\ samples} = \frac{100 + \ 50}{165} = 0.91$$

When the classification task consists on more than two classes, a multiclass problem, a similar definition of the confusion matrix can be extended from the binary problem. Considering a certain observation C_k, the True positive part of the matrix is placed in the exact point where the column and the row of this certain observation are crossed, i.e, when the predicted label is equal to the true label. The False positives samples are placed along the column C_k for all the rows $C_0, ..., C_{k-1}, Ck + 1, ..., C_n$ which refers to all the samples that have been misclassified with the class C_k. The False negatives are, however, all the samples that originally are labelled with C_k tag but have been wrongly categorized with $C_0, ..., C_{k-1}, Ck + 1, ..., C_n$ classes. Finally, the True negative samples are distributed across all the other positions in the matrix. In figure 25, a good example for this explanation is shown.

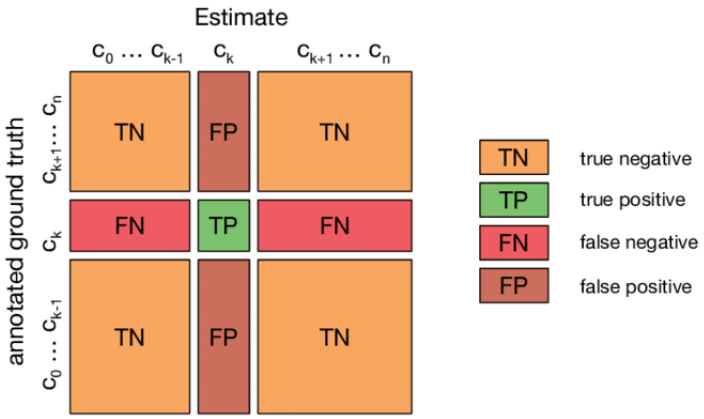


Fig. 25. Confusion matrix for a multiclass classification [102]

.2. k-Fold Cross Validation

The cross-validation technique consists in a resampling practice that is commonly used in order to evaluate machine learning algorithms when the dataset is not very large. It just depends on the parameter k which represents the amount of folds or groups the data is

going to be split into. This is the reason of the k-fold prefix. When the method is referred to in a situation in which the parameter is already fixed, for example, if $k = 5$, it can becomes named as 5-fold cross-validation.

The purpose of this procedure is to check the performance of a machine learning model on unseen data. This is for not evaluating with the data used for the training process. It is really used nowadays and allows to obtain a less biased estimation than if using other kind of techniques as just a simple train and test split [103]. The steps followed by the method are listed below [104].

1. The data is split in a random way into k folds. The value of this parameter can be chosen previously by running the algorithm for different options and picking the one with the best performance. However, a not very high value is usually chosen, keeping it in a range between 5 and 10.

2. Fit the model by using the data in the $k - 1$ folds as train set, and the other part in the k fold for the validation process. Collect the results for this configuration.

3. The process must be repeated until all folds have been used as the validation set. Then, the average and standard deviation of all the results can be computed as the metric of the model.

Sometimes, instead of dividing the whole set of data in k folds, it is first done a split into train and test sets. Then, the test set is excluded for a final measure and the train is split again with the k-Fold Cross Validation procedure. The concept of the technique is shown in figure 26.

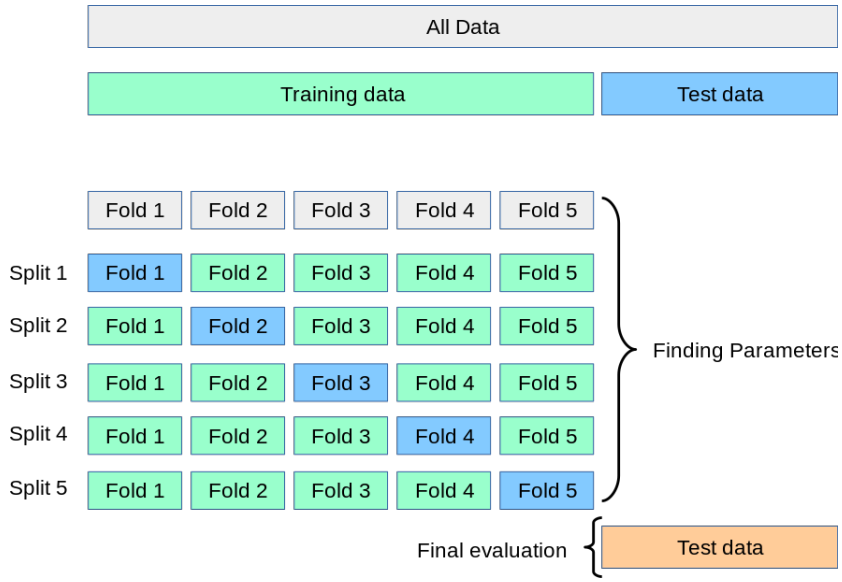


Fig. 26. K-fold cross-validation scheme. The data is first split into train and test, and then the train is split again with this method [105]

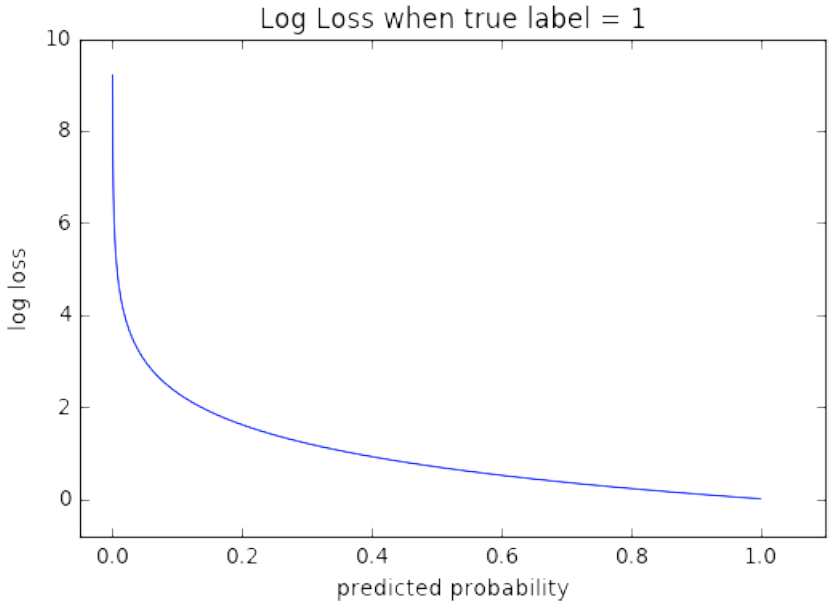


Fig. 27. Cross-entropy loss function when the true label is equals to 1. As the probability of the predicted class approaches to 1, the loss function tends to zero. However, if it the probability is closer to 0.0, then the loss function increases heavily [106].

.3. Categorical cross-entropy

In order to define this kind of loss function, it is necessary to consider the *softmax* definition. As it was already explained in **??**, this function takes a vector and represent its elements in the range (0,1) as probabilities, so all of the resulting values must sum up 1. In a multiclass classification, these are interpreted as class probabilities.

These output probabilities can be evaluated with the cross-entropy loss. This is a way of measuring the performance of a model in which the output are probabilities between 0 and 1. The loss increases if the probability differs largely from the actual value of the true label. A perfect system would have a loss value of 0 [106]. In figure 5.10, it is shown an example of the loss function when the true label is 1.

Its formula is defined for binary problems as follows:

$$CE = -(ylog(p) + (1 - y)log(1 - p))$$

For a multiclass case, a unique loss is computed for each class label for each sample, and then add the results as follows:

$$CE = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

Where M is then number of classes, y is a indication in binary format if the class c is the right prediction for this sample o and p is the estimation probability of c for o [106].

Then, the categorical cross-entropy is a combination of softmax and the cross-entropy loss.

.4. Spectrogram and Mel scale

In his natural conditions and how humans understand audio when we hear it, it can be interpreted as a succession of sounds that depend on time. Actually, what happens during a recording process is that the signal registered is nothing but the variation of the values that represent the changes in the acoustic pressure along time, what is usually called the amplitude of the signal. This is what we see when visualizing a raw audio file in its original form, i.e. in the time-domain [107], as in figure 28.

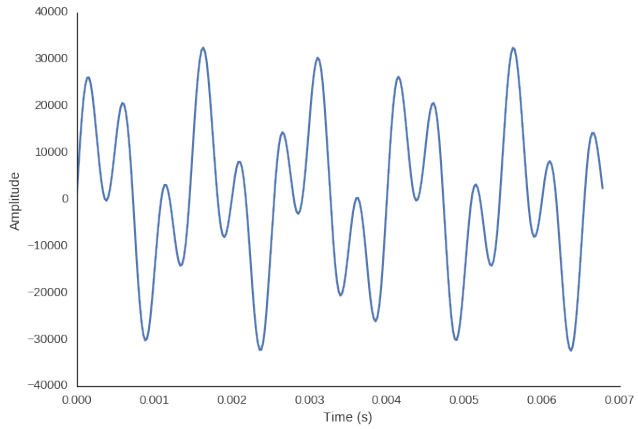


Fig. 28. Audio signal in time domain

However, in his natural form, audio data is not very useful for learning tasks. With the objective of representing audio data in another format so information can be extracted in a more practical way, it is normally converted to the frequency domain. The FT is the technique in charge of performing this conversion. Since in signal processing we work with discrete data, the algorithm is called Discrete Fourier Transform (DFT), which is essentially the same as FT but for discrete data. The most common algorithm to compute this is the one called Fast Fourier Transform (FFT), which performs the DFT in a much more efficient way [108]. The result of this method shows the values of the amplitude against the frequency axis. There is also another method to perform this operation and it is called the Short-Time Fourier Transform (STFT), which is the one used in this work.

The STFT can be defined as a sequence of FT s which are computed along a windowed signal. So, apart from computing the transformation, it also localizes in time this frequency information, given as a result a variation of the frequency along the time axis. Below, the formula of this operation is included:

$$X_{STFT} = \sum_{k=0}^{L-1} x[k]g[k-m]e^{-j2\pi nk/L}$$

$x[k]$ is the time signal of length L and $g[k]$ is the window that goes through all the signal. In figure 29, a visualization of the whole process is shown [109].

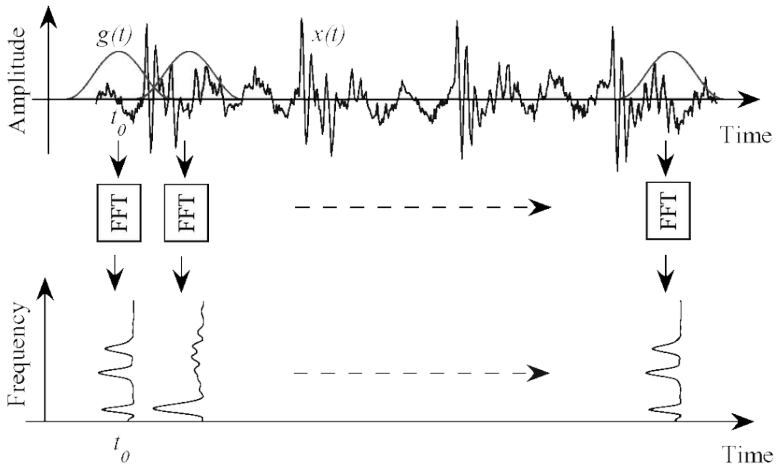


Fig. 29. STFT [110]

What is obtained from this computation after going through all the signal, it is a representation that depends on three magnitudes: time, frequency and the amplitude. This is known as the spectrogram of the signal. With this visualization we can see how the energy of the signal varies with time and also how it is placed in the different frequencies. In figure 30 it is shown how a typical representation of a spectrogram looks like.

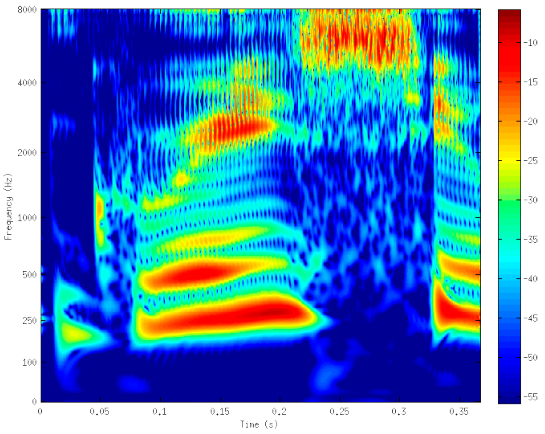


Fig. 30. Spectrogram of an audio signal. The x-axis corresponds to the time, the y-axis to the frequency and the colour of the plot to the amplitude by following the colour bar on the right.

When dealing with audio learning tasks a really important magnitude that usually is

employed is the Mel scale. This is the result of applying a non-linear transformation to the frequency scale. It has been used successfully in plenty of occasions because its understanding of sounds in a very similar way as how humans do [111]. This is done by considering the fact that we are more capable of differentiating little changes on pitch at low frequencies than at high frequencies. The following formula is the way to convert from frequency to mel scale [112]:

$$M(f) = 1127 \ln (1 + \frac{f}{700})$$

This same idea of applying the human understanding of sounds has also been used to generate the Mel-frequency Cepstrum Coefficients (MFCC), which have been widely used in speech problems. The first thing to understand in this field is that the sounds we produce are filtered somehow by the entire oral cavity. If a similar representation could be modelled in an artificial system, a more accurate depiction of the sound pronounced could be achieved. Actually, the envelope of the short time power spectrum of the signal can be considered as the shape of this filter and it can be reliably represented by the MFCC, so useful information can be deduced from them. This type of features have been extensively used for several fields such as automatic speech generation or speaker recognition [12].